Schemata of SMT problems

Vincent Aravantinos, Nicolas Peltier

Schemata of SMT-problems

Vincent Aravantinos and Nicolas Peltier

University of Grenoble (LIG, CNRS)

Abstract. A logic is devised for reasoning about iterated schemata of SMT problems. The satisfiability problem is shown to be undecidable for this logic, but we present a proof procedure that is sound, complete w.r.t. satisfiability and terminating for a precisely characterized class of problems. It is parameterized by an external procedure (used as a black box) for testing the satisfiability of ground instances of the schema in the considered theory (e.g. integers, reals etc.).

1 Introduction

In [1] a logic is defined for reasoning on schemata of propositional formulae. It extends standard propositional logic by using *indexed symbols* (e.g. p_0, $p_{\mathtt{i}}$, $p_{\mathtt{i}+1}$, etc.), *arithmetic parameters* (i.e. constant symbols interpreted as natural numbers) and *iterated connectives* such as $\bigvee_{\mathtt{i}=0}^{\mathtt{n}} p_{\mathtt{i}}$ or $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}} p_{\mathtt{i}}$ (where $\mathtt{n}$ denotes a *parameter*, not a fixed number) that can be viewed as formulae with bounded quantifiers $\exists \mathtt{i} \in [0, \mathtt{n}], p_{\mathtt{i}}$ and $\forall \mathtt{i} \in [0, \mathtt{n}], p_{\mathtt{i}}$. It is shown that the validity problem is undecidable when arbitrary indices and (linear) arithmetic expressions are considered. The problem is co-semi-decidable and decision procedures of "reasonable" complexity can be defined for some interesting classes (see [2] for details). A simple example is the following schema: $p_0 \wedge p_{\mathtt{n}+1} \wedge \bigwedge_{\mathtt{i}=0}^{\mathtt{n}} (p_{\mathtt{i}} \Leftrightarrow \neg p_{\mathtt{i}+1})$, that is satisfiable if and only if $\mathtt{n}$ is odd. This formula can be reduced into a propositional one by fixing the value of $\mathtt{n}$, e.g. for $\mathtt{n} \leftarrow 0$: $p_0 \wedge p_1 \wedge (p_0 \Leftrightarrow \neg p_1)$, or for $\mathtt{n} \leftarrow 1$: $p_0 \wedge p_2 \wedge (p_0 \Leftrightarrow \neg p_1) \wedge (p_1 \Leftrightarrow \neg p_2)$. A SAT-solver can determine whether the formula is satisfiable or unsatisfiable for a given value of $\mathtt{n}$ and a model can be found (if it exists) by enumerating all possible values ($n \leftarrow 0, 1, 2, \ldots$). However, proving that such a formula is unsatisfiable *for all values of* $\mathtt{n}$ (which is the case for instance if one adds the constraint $\mathtt{n} = 2 \times \mathtt{m}$) is much difficult, and usually requires to use some particular form of mathematical induction. The proof procedure described in [1] combines usual tableaux-based decomposition rules with lazy instantiation of the parameter and a loop detection mechanism that captures a restricted form of "descente infinie" induction reasoning ensuring completeness in some cases.

Our aim in this paper is to extend these results to schemata of (quantifier-free) SMT-problems (standing for **S**atisfiability **M**odulo **T**heory). Proving the unsatisfiability (or satisfiability) of a ground formula modulo some background theory is an essential problem in computer science, in particular for the automatic verification of complex systems. In software verification for example, the background theory can define data structures such as integers, arrays or lists.

These problems are known as $\mathcal{T}$-*decision problems* or more commonly, *SMT problems*, and the tools capable of solving these problems are known as $\mathcal{T}$-*decision procedures*, or *SMT solvers*. A lot of research has been devoted to the design of SMT solvers that are both efficient and scalable. A survey can be found in [3].

The schemata we consider in this paper may be seen as (countably infinite) families of SMT-problems, parameterized by a natural number $\mathtt{n}$. Both the signature of problems and the set of axioms may depend on $\mathtt{n}$. Consider for instance the following formula, representative of those arising in, e.g, verifying programs handling arrays: $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}} a_{\mathtt{i}+1} \succeq a_{\mathtt{i}} \wedge \bigwedge_{\mathtt{i}=0}^{\mathtt{n}} b_{\mathtt{i}+1} \preceq b_{\mathtt{i}} \wedge a_0 \succeq b_0 \wedge a_{\mathtt{n}+1} \preceq c \wedge b_{\mathtt{n}+1} \succ c$. It is not hard to see that this example is unsatisfiable. Again, by instantiating $\mathtt{n}$, say to 1, we get a ground formula: $a_1 \succeq a_0 \wedge a_2 \succeq a_1 \wedge b_1 \preceq b_0 \wedge b_2 \preceq b_1 \wedge a_0 \succeq b_0 \wedge a_2 \preceq c \wedge b_2 \succ c$. The satisfiability of this formula modulo, e.g., arithmetic can be tested by any SMT-solver. However proving that the original schema is unsatisfiable *for every* $\mathtt{n} \in \mathbb{N}$ is out of the scope of these tools. One can of course encode such a schema as a *non-ground* (i.e. with universal quantifier) SMT-problem, simply by considering $\mathtt{n}$ as a constant symbol of sort integer, by writing indices as arguments, and by replacing iterated connectives by quantifiers:

$$\begin{array}{rl} & \forall i,\, 0 \preceq \mathtt{i} \wedge \mathtt{i} \preceq \mathtt{n} \Rightarrow a(\mathtt{i}+1) \succeq a(\mathtt{i}) \\ \wedge & \forall i,\, 0 \preceq \mathtt{i} \wedge \mathtt{i} \preceq \mathtt{n} \Rightarrow b(\mathtt{i}+1) \preceq b(\mathtt{i}) \\ \wedge\, a(0) \succeq b(0) \wedge a(\mathtt{n}+1) \preceq c \wedge b(\mathtt{n}+1) \succ c & \end{array}$$

However, this is of no practical use since of course there is no complete and terminating procedure for solving non-ground SMT-problem. The heuristics that are used by SMT-solvers to handle quantifiers, although rather efficient and powerful in some cases, cannot handle such formulae. For instance the well-known SMT-solver Yices [6] that uses E-matching [5] for instantiating universally quantified variables fails to establish the unsatisfiability of this schema. Some complete techniques have been proposed for instantiating universal quantifiers [7] but they do not terminate in our case. Alternatively, indexed constant symbols can be modeled by arrays (with quantifiers on the indices), however the obtained formulae are again outside the known decidable classes [4, 8]. The reason is that the formulae obtained by encoding schemata of SMT-problems cannot, in general, be reduced to unsatisfiable ground formulae by *finitely* grounding the universally quantified variables: the logic is not compact and using mathematical induction is required. Our approach extends SMT-solvers with a limited form of mathematical induction.

The rest of the paper is structured as follows. In Section 2 we introduce the syntax and semantics of our logic and we show that the satisfiability problem is undecidable (even in cases in which purely propositional schemata are actually decidable). In Section 3 we devise a very general and abstract proof procedure that relies on semantic properties of the considered class of problems. In Section 4 we give concrete examples of classes satisfying the previous requirements, thus turning the abstract procedure in Section 3 into concrete decision procedures for these classes. Examples are provided in Section 5 and Section 6 briefly concludes the paper. Due to space restriction, the proofs are skipped to the Appendix.

2 Preliminaries

We define the logic of $\mathcal{T}$-*schemata*, where $\mathcal{T}$ is a theory (more precisely a class of interpretations) for which the satisfiability problem is assumed to be decidable.

2.1 Syntax

We consider terms built on a signature containing indexed constants and function symbols, where the indices are arithmetic expressions. We assume that the symbols are indexed by at most one index (e.g. $a_{\mathtt{i},\mathtt{j}}$ is forbidden) and that the expressions contain at most one occurrence of an arithmetic variable (e.g. $a_{\mathtt{i}+\mathtt{j}}$ and even $a_{\mathtt{i}+\mathtt{i}}$ are not allowed, but $f_{\mathtt{i}+1}(a_{\mathtt{i}})$ and $f_0(a_{\mathtt{i}})$ are)[1]. We also assume that the considered formulae contain a unique parameter, which is interpreted by a natural number. More formally:

Let $\mathtt{n}$ and $\mathtt{i}$ be two distinct symbols. $\mathtt{n}$ is the *parameter* and $\mathtt{i}$ is the *index variable*. The set of *index expressions* is $\{\mathtt{i}, \mathtt{i}+1\} \cup \{succ^k(0) \mid k \in \mathbb{N}\} \cup \{succ^k(\mathtt{n}) \mid k \in \mathbb{N}\}$. As usual, the expressions $succ^k(0)$ and $succ^k(\mathtt{n})$ (where $k \in \mathbb{N}$) are written k and $\mathtt{n}+k$ respectively.

Let $\mathtt{Sorts}$ denote a set of *sort symbols* (containing in particular a symbol $\mathtt{bool}$) and let $\mathcal{F}$ denote a set of *function symbols*, partitioned into two disjoint sets $\mathcal{F} = \mathcal{F}_{\mathrm{I}} \uplus \mathcal{F}_{\mathrm{NI}}$: the *indexed symbols* $\mathcal{F}_{\mathrm{I}}$ and the *non-indexed symbols* $\mathcal{F}_{\mathrm{NI}}$. Each symbol $f \in \mathcal{F}$ is mapped to a unique *profile* of the form $\mathtt{s}_1, \ldots, \mathtt{s}_k \to \mathtt{s}$, where $k \in \mathbb{N}$ and $\mathtt{s}_1, \ldots, \mathtt{s}_k, \mathtt{s} \in \mathtt{Sorts}$. This is written $f : \mathtt{s}_1, \ldots, \mathtt{s}_k \to \mathtt{s}$ or simply $f : \mathtt{s}$ if $k = 0$ (in this case f is a *constant*). If $\mathtt{s} = \mathtt{bool}$ then f is a *predicate symbol*. k is the *arity* of f. We assume that $\mathcal{F}_{\mathrm{NI}}$ contains in particular a symbol $\mathtt{true} : \mathtt{bool}$.

The set $T(\mathtt{s})$ of *terms of sort* $\mathtt{s}$ is the smallest set of expressions satisfying the following conditions:

- If $f : \mathtt{s}_1, \ldots, \mathtt{s}_k \to \mathtt{s}$ is a non-indexed function symbol and if $u_1, \ldots, u_k$ are terms of sort $\mathtt{s}_1, \ldots, \mathtt{s}_k$ respectively, then $f(u_1, \ldots, u_k)$ is a term of sort $\mathtt{s}$.
- If $f : \mathtt{s}_1, \ldots, \mathtt{s}_k \to \mathtt{s}$ is an indexed function symbol, if α is an index expression and if $u_1, \ldots, u_k$ are terms of sort $\mathtt{s}_1, \ldots, \mathtt{s}_k$ respectively, then $f_\alpha(u_1, \ldots, u_k)$ is a term of sort $\mathtt{s}$.

Note that, by construction, the only variable occurring in a term is $\mathtt{i}$ (there are no non-arithmetic variables).

For instance, if $\mathcal{F}_{\mathrm{I}} = \{a : \mathtt{elem}, \mathtt{f} : \mathtt{elem}, \mathtt{elem} \to \mathtt{elem}\}$ and $\mathcal{F}_{\mathrm{NI}} = \{b : \mathtt{elem}, \mathtt{p} : \mathtt{elem} \to \mathtt{bool}\}$ then $a_0, a_{\mathtt{n}}, a_{\mathtt{i}+1}, f_{\mathtt{n}+2}(a_0, b), f_0(a_{\mathtt{i}+2}, a_3)$ are terms of sort $\mathtt{elem}$ and $p(a_{\mathtt{i}+1})$ is a term of sort $\mathtt{bool}$. Terms such as $a_{\mathtt{i}+2}, a_{\mathtt{i}+\mathtt{n}}$ are not allowed (indeed, $\mathtt{i}+2$ and $\mathtt{i}+\mathtt{n}$ are not index expressions).

Now we define the syntax of formulae. For technical convenience, we assume that all formulae are in negative normal form. An *atom* is of the form $u \approx v$, where u and v are two terms of the same sort. An atom of the form $u \approx \mathtt{true}$ is

[1] Removing these conditions yields undecidable logics, even in the purely propositional case [1, 2], thus we prefer to add them immediately rather than defining a very general formalism that will have to be strongly restricted at a later stage (as done in [1]).

non-equational. A *literal* is of the form $u \approx v$ or $u \not\approx v$. For readability a literal $u \approx \mathtt{true}$ or $u \not\approx \mathtt{true}$ is simply written u or $\neg u$. $\mathtt{false}$ is a shorthand for $\neg\mathtt{true}$.

An *iteration body* is inductively defined as either a literal *not containing* $\mathtt{n}$ or a formula of the form $\phi \vee \psi$ or $\phi \wedge \psi$ where ϕ and ψ are iteration bodies. Finally, a *schema* is inductively defined as follows:

- A literal *not containing* $\mathtt{i}$ is a schema.
- If ϕ and ψ are schemata then $\phi \vee \psi$ and $\phi \wedge \psi$ are schemata.
- If ϕ is an iteration body containing $\mathtt{i}$ and α is an index expression of the form k or $\mathtt{n}+k$ (where $k \in \mathbb{N}$) then $\bigvee_{\mathtt{i}=0}^{\alpha} \phi$ and $\bigwedge_{\mathtt{i}=0}^{\alpha} \phi$ are schemata.

For readability, we will sometimes use the abbreviation $u \approx v \Rightarrow \psi$ (resp. $u \not\approx v \Rightarrow \psi$) for $u \not\approx v \vee \psi$ (resp. $u \approx v \vee \psi$).

$\mathfrak{S}$ denotes the set of all schemata. A schema is *iteration-free* iff it contains no iterated connective $\bigvee$ or $\bigwedge$ and *parameter-free* if it contains no occurrence of $\mathtt{n}$. A *sentence* is a schema that is both iteration-free and parameter-free. Such a schema may be viewed as a standard (quantifier-free) formula in the usual sense (with function symbols indexed by natural numbers), but we prefer not to use the word "formula" to avoid confusions.

For instance $\phi_1 : \bigvee_{\mathtt{i}=0}^{\mathtt{n}} (a_{\mathtt{i}+1} \approx f(b_{\mathtt{i}}) \wedge b_{\mathtt{i}+1} \approx g(a_{\mathtt{i}}))$, $\phi_2 : a_{\mathtt{n}+1} \approx f(b_{\mathtt{n}}) \wedge b_{\mathtt{n}+1} \approx g(a_{\mathtt{n}})$, $\phi_3 : \bigvee_{\mathtt{i}=0}^{3} (a_{\mathtt{i}+1} \approx f(b_{\mathtt{i}}) \wedge b_{\mathtt{i}+1} \approx g(a_{\mathtt{i}}))$ and $\phi_4 : a_1 \approx f(b_0) \wedge b_1 \approx g(a_0)$ are schemata. ϕ_2 and ϕ_4 are iteration-free, ϕ_3 and ϕ_4 are parameter-free and ϕ_4 is a sentence.

An *expression* may be a term, a vector of terms, an iteration body or a schema. It is *ground* if it contains no occurrence of $\mathtt{i}$ (notice that it *may contain* the parameter n) and *non-indexed* if it contains no indexed symbols (by definition all non-indexed expressions are ground).

Let α be a ground index expression. If ϕ is an iteration body then $\phi\{\mathtt{i} \leftarrow \alpha\}$ denotes the iteration-free schema obtained from ϕ by replacing all occurrences of $\mathtt{i}$ by α. If ϕ is a schema or an index expression, then $\phi\{\mathtt{n} \leftarrow \alpha\}$ denotes the schema or index expression obtained from ϕ by replacing all occurrences of $\mathtt{n}$ by α.

If ϕ is an iteration-free schema or an iteration body, we denote by $\mathit{Ind}(\phi)$ the set of index expressions occurring in ϕ: $\mathit{Ind}(u_\alpha) \stackrel{\text{def}}{=} \{\alpha\}$, $\mathit{Ind}(u \approx v) = \mathit{Ind}(u \not\approx v) = \mathit{Ind}(u) \cup \mathit{Ind}(v)$, $\mathit{Ind}(\phi \star \psi) \stackrel{\text{def}}{=} \mathit{Ind}(\phi) \cup \mathit{Ind}(\psi)$ if $\star \in \{\vee, \wedge\}$.

2.2 Semantics

The semantics is straightforwardly defined. The only difference with first-order logic is that the parameter must be interpreted by a natural number and that the index variable ranges over $\mathbb{N}$. More precisely, a *schema interpretation* (or *interpretation* for short) I is a function mapping $\mathtt{n}$ to a natural number $\langle\mathtt{n}\rangle^I$, mapping each sort $s \in \mathtt{Sorts}$ to a non-empty set $\langle s\rangle^I$, mapping each non-indexed function symbol $f : \mathtt{s}_1, \ldots, \mathtt{s}_k \to \mathtt{s}$ to a function $\langle f\rangle^I : \langle\mathtt{s}_1\rangle^I, \ldots, \langle\mathtt{s}_k\rangle^I \to \langle\mathtt{s}\rangle^I$ and mapping each indexed function symbol $f : \mathtt{s}_1, \ldots, \mathtt{s}_k \to \mathtt{s}$ to a family of functions $\langle f\rangle^I_l : \langle\mathtt{s}_1\rangle^I, \ldots, \langle\mathtt{s}_k\rangle^I \to \langle\mathtt{s}\rangle^I$ (where $l \in \mathbb{N}$). The function $x \mapsto \langle x\rangle^I$ is then extended to any *ground* term or atom and to any schema as follows:

- If α is an index expression, then $\langle\alpha\rangle^I \stackrel{\text{def}}{=} \alpha\{\mathtt{n} \leftarrow \langle\mathtt{n}\rangle^I\}$. Notice that $\langle\alpha\rangle^I$ is then equivalent to a natural number.
- $\langle f(v_1,\ldots,v_k)\rangle^I \stackrel{\text{def}}{=} \langle f\rangle^I(\langle v_1\rangle^I,\ldots,\langle v_k\rangle^I)$.
- $\langle f_\alpha(v_1,\ldots,v_k)\rangle^I \stackrel{\text{def}}{=} \langle f\rangle^I_{\langle\alpha\rangle^I}(\langle v_1\rangle^I,\ldots,\langle v_k\rangle^I)$.
- $\langle\bigvee_{\mathtt{i}=0}^{\alpha}\phi\rangle^I \stackrel{\text{def}}{=} \mathtt{true}$ if there exists $l \in [0,\langle\alpha\rangle^I]$ such that $\langle\phi\{\mathtt{i}\leftarrow l\}\rangle^I = \mathtt{true}$.
- $\langle\bigwedge_{\mathtt{i}=0}^{\alpha}\phi\rangle^I \stackrel{\text{def}}{=} \mathtt{true}$ if for all $l \in [0,\langle\alpha\rangle^I]$ we have $\langle\phi\{\mathtt{i}\leftarrow l\}\rangle^I = \mathtt{true}$.

We omit the definitions for the symbols $\approx, \not\approx, \vee, \wedge$, which are standard. If ϕ is a schema, we write $I \models \phi$ iff $\langle\phi\rangle^I = \mathtt{true}$. In this case, I is a *model* of ϕ and ϕ is *satisfiable*.

Usually, satisfiability is tested w.r.t. a particular class of interpretations, in which the semantics of some of the symbols is fixed (for instance the sort symbol `int` is interpreted as $\mathbb{Z}$ and $+$ is interpreted as the addition). Let $\mathcal{T}$ be a class of interpretations. ϕ is *$\mathcal{T}$-satisfiable* iff there exists $I \in \mathcal{T}$ such that $I \models \phi$. Two schemata ϕ and ψ are *$\mathcal{T}$-equivalent* iff we have $I \models \phi \Leftrightarrow I \models \psi$ for every interpretation $I \in \mathcal{T}$ and *$\mathcal{T}$-sat-equivalent* iff ϕ and ψ are both $\mathcal{T}$-satisfiable or both $\mathcal{T}$-unsatisfiable. We assume that there exists an algorithm for checking whether a given sentence (i.e. a schema without iterated connective and without parameter) is $\mathcal{T}$-satisfiable or not.

A function f is *non-built-in* if its interpretation is arbitrary, i.e. for every interpretation $I \in \mathcal{T}$, the interpretation obtained from I by changing only the interpretation of f is also in $\mathcal{T}$. We assume that every indexed symbol is non-built-in (i.e. the only symbols whose interpretation is fixed are non-indexed).

Note that if I is an interpretation and α is a ground expression, then by definition $I \circ \{\mathtt{n} \mapsto \alpha\}$ is also an interpretation. $I \circ \{\mathtt{n} \mapsto \alpha\}$ and I coincide on every symbol, except on $\mathtt{n}$. If $\alpha \in \mathbb{N}$ then $\langle\mathtt{n}\rangle^{I\circ\{\mathtt{n}\mapsto\alpha\}} = \alpha$ and otherwise $\langle\mathtt{n}\rangle^{I\circ\{\mathtt{n}\mapsto\alpha\}} = \langle\alpha\rangle^I$.

Proposition 1. *For every parameter-free schema ϕ one can compute a sentence that is $\mathcal{T}$-equivalent to ϕ. Thus the $\mathcal{T}$-satisfiability problem is decidable for parameter-free schemata.*

As usual in SMT problems, we shall assume that the schemata are *flattened*, i.e. for every term of the form $f(u_1,\ldots,u_k)$ occurring in the schema (where f is possibly indexed) $u_1,\ldots,u_k$ are (possibly indexed) constant symbols. This is not restrictive, for instance a term of the form $f(g(a_\mathtt{i}), g_{\mathtt{i}+1}(a_0))$ can be replaced by $f(b_\mathtt{i}, c_\mathtt{i})$, where the axioms $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}} b_\mathtt{i} \approx g(a_\mathtt{i})$ and $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}} c_\mathtt{i} \approx g_{\mathtt{i}+1}(a_0)$ are added to the schema.

2.3 Extensions of the language

Several extensions of this basic language can be considered. We did not include them in the previous definitions because they do not increase the expressive power, but for readability we shall sometimes use them in the following.

- **Inequality tests.** Atoms of the form $\mathtt{i} \leq k$ (where $k \in \mathbb{N}$) can be added in iteration bodies. This does not increase the expressive power since such atoms can be equivalently replaced by atoms of the form $p_\mathtt{i}^{\leq k}$, where $p^{\leq k}$

is a fresh constant symbol of sort `bool` (depending on k), defined by the following axioms: $p_0^{\leq k} \wedge \cdots \wedge p_k^{\leq k} \wedge \neg p_{k+1}^{\leq k} \wedge \bigwedge_{\mathtt{i}=0}^{\mathtt{n}}(p_{\mathtt{i}+1}^{\leq k} \Rightarrow p_{\mathtt{i}}^{\leq k})$.

- **Arbitrary lower bounds.** Iterations whose lower bound is distinct from 0 can easily be expressed using the previous atoms: $\bigwedge_{\mathtt{i}=k}^{\mathtt{n}} \phi$ is written $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}}(\mathtt{i} \leq k-1 \vee \phi)$ (if $k > 0$).
- **Arbitrary translations.** Terms of the form $a_{\mathtt{i}+k}$ can also be considered, where $k > 1$. Indeed, such a term can be replaced by a fresh constant symbol $a_{\mathtt{i}}^{+k}$, where a^{+k} is defined by the following axioms: $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}+k}\left(a_{\mathtt{i}}^{+0} \approx a_{\mathtt{i}} \wedge a_{\mathtt{i}}^{+1} \approx a_{\mathtt{i}+1}^{+0} \wedge \cdots \wedge a_{\mathtt{i}}^{+k} \approx a_{\mathtt{i}+1}^{+k-1}\right)$.
- **Additional parameters.** Inequalities of the form $\mathtt{i} \leq \mathtt{m}$ where $\mathtt{m}$ is an additional parameter interpreted as an element of $[0, \mathtt{n}]$ can be encoded by atoms $p_{\mathtt{i}}^{\leq \mathtt{m}}$ defined by the following axioms: $\neg p_{\mathtt{n}+1}^{\leq \mathtt{m}} \wedge p_0^{\leq \mathtt{m}} \wedge \bigwedge_{\mathtt{i}=0}^{\mathtt{n}}(p_{\mathtt{i}+1}^{\leq \mathtt{m}} \Rightarrow p_{\mathtt{i}}^{\leq \mathtt{m}})$. Then $\mathtt{i} \approx \mathtt{m}$ can be defined using $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}}(p_{\mathtt{i}}^{=\mathtt{m}} \Leftrightarrow p_{\mathtt{i}}^{\leq \mathtt{m}} \wedge \neg p_{\mathtt{i}+1}^{\leq \mathtt{m}})$ and a disequality $\mathtt{m} \not\approx \mathtt{k}$ can be tested by the schema: $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}}(\neg p_{\mathtt{i}}^{=\mathtt{m}} \vee \neg p_{\mathtt{i}}^{=\mathtt{k}})$.
- **Using the parameter in iteration bodies.** A term of the form $a_{\mathtt{n}}$ can be replaced by a fresh non-indexed constant b, with the axiom: $b \approx a_{\mathtt{n}}$.

2.4 Undecidability

The next theorem states that the considered logic is undecidable in general.

Theorem 1. *The satisfiability problem is undecidable for $\mathfrak{S}$.*

This result does *not* follow from the undecidability results in [1] or [2] (for propositional schemata) because the schemata considered here are much more restricted. The satisfiability problem is actually decidable if we restrict to propositional formulae (see Section 4.1). Intuitively, even though the language is sufficiently restricted to obtain decidability in the non-equational case, the equational part of the language adds enough power to "retrieve" the undecidability. This shows that the extension of schemata to SMT-problems is a difficult task.

3 Proof procedure

We define a proof procedure for testing the satisfiability of schemata that is sound and complete w.r.t. satisfiability. We show that, under some particular *semantic* conditions (depending both on the theory $\mathcal{T}$ and on the considered class of schemata), this procedure can be turned into a decision procedure.

3.1 Enumerating interpretations

We first define a semi-decision procedure for schemata. It is very simple but sufficient for our purposes. It is parameterized by a *simplification function* which is a function replacing a schema by a set of schemata (interpreted as a disjunction) in such a way that satisfiability is preserved.

Definition 1. *Let ϕ be a schema and let Ψ be a set of schemata. We write $\phi \rightsquigarrow \Psi$ iff the following conditions hold:*

1. *For every $I \in \mathcal{T}$, if $I \models \phi$ then there exists $\psi \in \Psi$ such that $I \models \psi$.*

2. *For every $I \in \mathcal{T}$, if there exists $\psi \in \Psi$ such that $I \models \psi$ then there exists an interpretation $J \in \mathcal{T}$ such that $J \models \phi$ and $\langle \mathtt{n} \rangle^J = \langle \mathtt{n} \rangle^I$.*

For instance, we have $\phi \vee \psi \rightsquigarrow \{\phi, \psi\}$, $(\phi \vee \phi') \wedge \psi \rightsquigarrow \{\phi \wedge \psi, \phi' \wedge \psi\}$, or $p_0 \wedge \phi \rightsquigarrow \{\phi\}$ if the indexed predicate symbol p does not occur in ϕ. We also have $\bigvee_{\mathtt{i}=0}^{\mathtt{n}+1} \phi \rightsquigarrow \{\bigvee_{\mathtt{i}=0}^{\mathtt{n}} \phi, \phi\{\mathtt{i} \leftarrow \mathtt{n}+1\}\}$. However we have $\neg p_0 \wedge p_{\mathtt{n}} \not\rightsquigarrow \{\mathtt{true}\}$ or $\neg p_0 \wedge \bigvee_{\mathtt{i}=0}^{\mathtt{n}} p_{\mathtt{i}} \not\rightsquigarrow \{\mathtt{true}\}$ (although $\neg p_0 \wedge p_{\mathtt{n}}$, $\neg p_0 \wedge \bigvee_{\mathtt{i}=0}^{\mathtt{n}} p_{\mathtt{i}}$ and $\mathtt{true}$ are $\mathcal{T}$-sat-equivalent). Notice that if $\phi \rightsquigarrow \Psi$ then ϕ is $\mathcal{T}$-sat-equivalent to the disjunction of the schemata in Ψ. Furthermore, if ϕ and the disjunction of the schemata in Ψ are equivalent then obviously $\phi \rightsquigarrow \Psi$.

Definition 2. *A* simplification function *is a total function $\Gamma : \mathfrak{S} \to 2^{\mathfrak{S}}$ such that for every $\phi \in \mathfrak{S}$, $\phi \rightsquigarrow \Gamma(\phi)$.*

As explained in the Introduction, a trivial way to construct a model of a schema ϕ (if it exists) is to enumerate all the possible values for $\mathtt{n}$ and then test the $\mathcal{T}$-satisfiability of the obtained sentences. Definition 3 formalizes this idea in a way that will be convenient for our purpose. We enumerate all possible instances of ϕ by instantiating recursively $\mathtt{n}$ by $\mathtt{n}+1$. A given simplification function Γ is systematically applied to the instantiated schemata:

Definition 3. *Let Γ be a simplification function. The* Γ-expansion *of a schema $\phi \in \mathfrak{S}$ is the set of schemata $\mathcal{E}_\Gamma(\phi)$ inductively built as follows:*

1. $\phi \in \mathcal{E}_\Gamma(\phi)$.
2. *If $\psi \in \mathcal{E}_\Gamma(\phi)$ then $\Gamma(\psi\{\mathtt{n} \leftarrow \mathtt{n}+1\}) \subseteq \mathcal{E}_\Gamma(\phi)$.*

Theorem 2. *Let Γ be a simplification function and let ϕ be a schema. ϕ is $\mathcal{T}$-satisfiable iff $\mathcal{E}_\Gamma(\phi)$ contains a schema ψ such that $\psi\{\mathtt{n} \leftarrow 0\}$ is $\mathcal{T}$-satisfiable.*

Theorem 2 implies that $\mathcal{T}$-satisfiability is semi-decidable for schemata in $\mathfrak{S}$. Indeed, to test whether $\phi \in \mathfrak{S}$ is $\mathcal{T}$-satisfiable, it suffices to construct the Γ-expansion $\mathcal{E}_\Gamma(\phi)$ of ϕ (using a straightforward simplification function, e.g. $\Gamma(\phi) = \{\phi\}$). By Definition 3, $\mathcal{E}_\Gamma(\phi)$ is recursively enumerable. By Theorem 2, ϕ is $\mathcal{T}$-satisfiable iff a schema ψ such that $\psi\{\mathtt{n} \leftarrow 0\}$ is $\mathcal{T}$-satisfiable is eventually obtained. The satisfiability of $\psi\{\mathtt{n} \leftarrow 0\}$ is decidable by Proposition 1. Of course, as such, this algorithm is very inefficient and seldom terminates (when the schema at hand is unsatisfiable): its efficiency and termination essentially depend on the choice of the simplification function.

The next definition states a condition on Γ ensuring that all the schemata in $\mathcal{E}_\Gamma(\phi)$ remain in a given class.

Definition 4. *Let $\mathcal{C}$ be a class of schemata. A simplification function Γ is* $\mathcal{C}$-preserving *iff $\phi \in \mathcal{C} \Rightarrow \Gamma(\phi\{\mathtt{n} \leftarrow \mathtt{n}+1\}) \subseteq \mathcal{C}$.*

Proposition 2. *Let $\mathcal{C}$ be a class of schemata. Let $\phi \in \mathcal{C}$ and let Γ be a $\mathcal{C}$-preserving simplification function. $\mathcal{E}_\Gamma(\phi) \subseteq \mathcal{C}$.*

3.2 Terminaison

We define a simplification function ensuring terminaison (for a particular class of schemata) of the proof procedure defined in Section 3. The intuitive idea is the following: the Γ-expansion of a given schema ϕ is infinite in general, since the recursive replacement of $\mathtt{n}$ by $\mathtt{n}+1$ creates schemata with increasingly deep index expressions. For instance from $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}}(p_{\mathtt{i}} \Rightarrow p_{\mathtt{i}+1})$ one gets $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}+1}(p_{\mathtt{i}} \Rightarrow p_{\mathtt{i}+1})$, $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}+2}(p_{\mathtt{i}} \Rightarrow p_{\mathtt{i}+1})$, ... A first step towards termination would be to have the iteration $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}}(p_{\mathtt{i}} \Rightarrow p_{\mathtt{i}+1})$ instead of this infinite set of iterations. This is easily obtained by *unfolding* the previous iterations (i.e. taking out the ranks $\mathtt{n}+1$, $\mathtt{n}+2$, etc.). However we are of course left with the new formulae introduced by those unfoldings. For instance, in the same example, one would get $p_{\mathtt{n}+1} \Rightarrow p_{\mathtt{n}+2}$, $p_{\mathtt{n}+2} \Rightarrow p_{\mathtt{n}+3}$, etc. One way to obtain termination is if we are able to somehow simplify those new formulae (of course this simplification depends on the considered theory $\mathcal{T}$) so that they belong to a finite set. This goal can be reached, in particular, if the indices of the involved atoms are restricted to be lower than $\mathtt{n}+k$ for some fixed $k \in \mathbb{N}$. It is actually sufficient to consider $k=1$, which leads to the following notion:

Definition 5. *A schema is* $\mathtt{n}$-elementary *if it contains no index of the form* $\mathtt{n}+k$ *where* $k>1$.

The major problem is, of course, to transform the schemata into $\mathtt{n}$-elementary ones (preserving $\mathcal{T}$-sat-equivalence). This may be done, *in some particular cases*, by using decomposition and simplification rules. In the previous example, the unfolding yields: $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}}(p_{\mathtt{i}} \Rightarrow p_{\mathtt{i}+1}) \wedge (p_{\mathtt{n}+1} \Rightarrow p_{\mathtt{n}+2})$. Then in order to eliminate all the indices greater than $\mathtt{n}+1$, we only have to eliminate $p_{\mathtt{n}+2}$ which can be done in this simple case by considering all the possible values for $p_{\mathtt{n}+2}$ (`true` or `false`). This yields the disjunction of the following schemata: $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}}(p_{\mathtt{i}} \Rightarrow p_{\mathtt{i}+1}) \wedge \neg p_{\mathtt{n}+1}$ (if $p_{\mathtt{n}+2}$ is false) and $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}}(p_{\mathtt{i}} \Rightarrow p_{\mathtt{i}+1})$ (if $p_{\mathtt{n}+2}$ is true).

Of course this case is an easy one, since the domain of the constant symbols is finite (thus every constant can be eliminated, if needed, by instantiation). But consider the case: $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}+1}(a_{\mathtt{i}} \approx f(a_{\mathtt{i}+1}))$. Here the unfolding yields the literal $a_{\mathtt{n}+1} \approx f(a_{\mathtt{n}+2})$. Since the domain is, a priori, not finite, the same technique cannot apply. Thus the ability to eliminate $a_{\mathtt{n}+2}$ depends on the theory $\mathcal{T}$: if, for instance, f is the successor function on $\mathbb{N}$ then it suffices to state that $a_{\mathtt{n}+1} \succ 0$.

To ensure that non-$\mathtt{n}$-elementary literals can always be eliminated, we will have to impose additional conditions on the class of interpretations $\mathcal{T}$ and on the considered schemata. To restrict the class of schemata we shall actually impose conditions on the literals occurring in it:

Definition 6. *A* frame $\mathfrak{L}$ *is a* finite *set of literals such that for every* $\lambda \in \mathfrak{L}$, *the two following conditions hold:*

1. $\lambda\{\mathtt{i} \leftarrow \mathtt{n}+1\} \in \mathfrak{L}$.
2. *If* λ *is* $\mathtt{n}$*-elementary then* $\lambda\{\mathtt{n} \leftarrow \mathtt{n}+1\} \in \mathfrak{L}$.

A schema ϕ *is* $\mathfrak{L}$-dominated *if every literal occurring in* ϕ *(both in iteration bodies and outside iterations) is in* $\mathfrak{L}$.

Those conditions are useful to ensure that a class of $\mathtt{n}$-elementary schemata is closed under replacement of $\mathtt{n}$ by $\mathtt{n}+1$ and unfolding of the iterations.

Example 1. The following set $\mathfrak{L}$ is a frame: $\{f(a_\mathtt{i}) \approx b_\mathtt{i}, f(a_\mathtt{i}) \not\approx g(b_{\mathtt{i}+1}), f(a_n) \approx b_n, f(a_n) \not\approx g(b_{n+1}), f(a_{n+1}) \approx b_{n+1}, f(a_{n+1}) \not\approx g(b_{n+2}), f(a_{n+2}) \approx b_{n+2}\}$. The literals $f(a_\mathtt{i}) \approx b_\mathtt{i}$, $f(a_\mathtt{i}) \not\approx g(b_{\mathtt{i}+1})$, $f(a_n) \approx b_n$, $f(a_n) \not\approx g(b_{n+1})$ and $f(a_{n+1}) \approx b_{n+1}$, are $\mathtt{n}$-elementary, $f(a_{n+1}) \not\approx g(b_{n+2})$ and $f(a_{n+2}) \approx b_{n+2}$ are not.

$\phi : (\bigvee_{\mathtt{i}=0}^{\mathtt{n}} f(a_\mathtt{i}) \approx b_\mathtt{i}) \wedge f(a_n) \not\approx g(b_{n+1})$ and $\psi : (\bigwedge_{\mathtt{i}=0}^{\mathtt{n}} f(a_\mathtt{i}) \not\approx g(b_{\mathtt{i}+1})) \vee f(a_{n+2}) \approx b_{n+2}$ are $\mathfrak{L}$-dominated. ϕ is $\mathtt{n}$-elementary, ψ is not.

The definition of the simplification function is divided into two steps: *unfolding* and *decomposition*.

Unfolding The first step simply aims at unfolding iterations, for instance by replacing $\bigvee_{\mathtt{i}=0}^{\mathtt{n}+1} \phi$ by $\bigvee_{\mathtt{i}=0}^{\mathtt{n}} \phi \vee \phi\{\mathtt{i} \leftarrow \mathtt{n}+1\}$. Obviously this is possible only if the lower bound of the iteration is strictly lower than the upper bound.

Definition 7. *If ϕ is a schema, we denote by unfold(ϕ) the schema obtained from ϕ by replacing every subschema of the form $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}+k} \psi$ or $\bigvee_{\mathtt{i}=0}^{\mathtt{n}+k} \psi$ occurring in ϕ such that $k > 0$ by (respectively): $(\bigwedge_{\mathtt{i}=0}^{\mathtt{n}} \psi) \wedge \psi\{\mathtt{i} \leftarrow \mathtt{n}+1\} \wedge \cdots \wedge \psi\{\mathtt{i} \leftarrow \mathtt{n}+k\}$ and $(\bigvee_{\mathtt{i}=0}^{\mathtt{n}} \psi) \vee \psi\{\mathtt{i} \leftarrow \mathtt{n}+1\} \vee \cdots \vee \psi\{\mathtt{i} \leftarrow \mathtt{n}+k\}$.*

The unfolding transformation does not affect the semantics of the considered schema. It is useful only to extract (when possible) the last operands of the iterations in order to pave the way for the elimination of the terms with greatest indices, which is done in the next subsection.

Decomposing schemata The second step is more complex. It aims at eliminating, in a schema ϕ, all the symbols whose index is greater than $\mathtt{n}+1$. This is the crucial part of our procedure, since the elimination of those symbols will ensure that only finitely many distinct schemata can be generated, hence that $\mathcal{E}_\Gamma(\phi)$ is finite. We now introduce the conditions on $\mathfrak{L}$ and $\mathcal{T}$ that ensure that the elimination of literals whose indices are strictly greater than $\mathtt{n}+1$ is feasible.

If I, J are two interpretations, we write $I \sim^{\mathfrak{L}} J$ iff I and J coincide on every literal obtained from a literal in $\mathfrak{L}$ by replacing $\mathtt{i}$ by a natural number lower or equal to $\mathtt{n}$. More precisely, $I \sim^{\mathfrak{L}} J$ if I and J coincide on $\mathtt{n}$ and on every sort symbol in $\mathtt{Sorts}$, and if for every literal $\lambda \in \mathfrak{L}$ containing $\mathtt{i}$ and for every $k \in [0, \langle \mathtt{n} \rangle^I]$ we have $\langle \lambda\{\mathtt{i} \leftarrow k\} \rangle^I = \langle \lambda\{\mathtt{i} \leftarrow k\} \rangle^J$.

Definition 8. *A frame $\mathfrak{L}$ is* stably decomposable, *relatively to a function $\Delta : \mathfrak{S} \to \mathfrak{S}$, iff for all* ground *non-$\mathtt{n}$-elementary literals $\lambda_1, \ldots, \lambda_k \in \mathfrak{L}$ the following conditions hold:*

- *$\Delta(\lambda_1 \wedge \cdots \wedge \lambda_k)$ is a boolean combination of ground $\mathtt{n}$-elementary literals in $\mathfrak{L}$.*
- *For every interpretation I, $I \models \Delta(\lambda_1 \wedge \cdots \wedge \lambda_k)$ iff there exists an interpretation J such that $J \sim^{\mathfrak{L}} I$ and $J \models \lambda_1 \wedge \cdots \wedge \lambda_k$.*

In what follows, we assume the existence of a frame $\mathfrak{L}$ and of a function Δ s.t. $\mathfrak{L}$ is stably decomposable w.r.t. Δ. Both depend on the theory $\mathcal{T}$. Thus, applying our method to a theory $\mathcal{T}$ requires that $\mathcal{T}$ be accompanied with a frame $\mathfrak{L}$ and a function Δ. We shall provide in Section 4 some examples of such frames and functions depending on $\mathcal{T}$.

We now define the simplification function. It is defined by means of a tableaux calculus, using the usual propositional decomposition rules. These rules are restricted to apply only on non-$\mathtt{n}$-elementary schemata. The goal is to decompose the schema in order to get rid of all non-$\mathtt{n}$-elementary literals occurring at non-root level. Then a new rule is defined, the so-called *Elimination rule*, in order to eliminate non-$\mathtt{n}$-elementary literals at root level, by taking advantage of the existence of a function Δ satisfying the conditions of Definition 8.

A *branch* is a conjunction of schemata and a *tableau* is a set of branches. As usual, tableaux are constructed using a set of *expansion rules* that are written in the form: $\dfrac{\mathcal{S}}{\mathcal{S}_1 \mid \ldots \mid \mathcal{S}_k}$ meaning that a branch that is of the form $\mathcal{S} \wedge \mathcal{S}'$ (up to the AC-properties of the connective $\wedge$) is deleted from the tableau and replaced by the k branches $\mathcal{S}_1 \wedge \mathcal{S}', \ldots, \mathcal{S}_k \wedge \mathcal{S}'$. If $k = 0$ the rule simply deletes (or *closes*) the branch. This is written $\dfrac{\mathcal{S}}{\bot}$. Initially, the tableau contains only one branch, defined by the schema at hand. We denote by ρ the following set of expansion rules:

$\vee$-Elimination: $\dfrac{\phi \vee \psi}{\phi \mid \psi}$ If $\phi \vee \psi$ is not $\mathtt{n}$-elementary.

Closure: $\dfrac{\phi \wedge \neg\phi}{\bot}$

Elimination: $\dfrac{\lambda_1 \wedge \cdots \wedge \lambda_k}{\Delta(\lambda_1 \wedge \cdots \wedge \lambda_k)}$ If $\{\lambda_1, \ldots, \lambda_k\} \subseteq \mathfrak{L}$ is the set of *all* the non-$\mathtt{n}$-elementary literals occurring in the branch.

We do not need a specific rule for the connective $\wedge$ since branches are considered as conjunctions (thus the $\wedge$-rule is implicitly replaced by the associativity of $\wedge$).

Proposition 3. *The non-deterministic application of the rules in ρ terminates on any schema.*

For every schema ϕ, we denote by $\rho^*(\phi)$ an arbitrarily chosen normal form of the tableau $\{\phi\}$ by the rules in ρ. Since tableaux are defined as sets of schemata (conjunctions), $\rho^*(\phi)$ is a set of (irreducible) schemata (i.e. the leaves).

Example 2. Let $\phi = \{(a_{\mathtt{n}+2} \succeq 0 \wedge a_{\mathtt{n}+2} \approx b_{\mathtt{n}+1} \vee p_{\mathtt{n}+1} \vee p_{\mathtt{n}}) \wedge \neg p_{\mathtt{n}+1} \wedge (p_{\mathtt{n}} \vee q_{\mathtt{n}+1})\}$. The application of the rules $\vee$-Elimination and Closure yields the two following branches:

$$a_{\mathtt{n}+2} \succeq 0 \wedge a_{\mathtt{n}+2} \approx b_{\mathtt{n}+1} \wedge \neg p_{\mathtt{n}+1} \wedge (p_{\mathtt{n}} \vee q_{\mathtt{n}+1})$$

and

$$p_{\mathtt{n}} \wedge \neg p_{\mathtt{n}+1} \wedge (p_{\mathtt{n}} \vee q_{\mathtt{n}+1})$$

Notice that the schema $(p_{\mathtt{n}} \vee q_{\mathtt{n}+1})$ is not decomposed, because it is $\mathtt{n}$-elementary. The second branch contains no non-$\mathtt{n}$-elementary schema hence is irreducible. The

non-$\mathtt{n}$-elementary conjuncts in the first branch are $a_{\mathtt{n}+2} \succeq 0$ and $a_{\mathtt{n}+2} \approx b_{\mathtt{n}+1}$. The rule Elimination applies, and the function Δ replaces these conjuncts by some $\mathcal{T}$-satequivalent conjunction of $\mathtt{n}$-elementary literals. In this case, it is intuitively obvious that we should take: $\Delta(a_{\mathtt{n}+2} \succeq 0 \wedge a_{\mathtt{n}+2} \approx b_{\mathtt{n}+1}) = b_{\mathtt{n}+1} \succeq 0$ (see Section 4 for the formal definition). Thus $\rho^*(\phi) = \{b_{\mathtt{n}+1} \succeq 0 \wedge \neg p_{\mathtt{n}+1} \wedge (p_{\mathtt{n}} \vee q_{\mathtt{n}+1}), p_{\mathtt{n}} \wedge \neg p_{\mathtt{n}+1} \wedge (p_{\mathtt{n}} \vee q_{\mathtt{n}+1})\}$.

Let $\mathfrak{S}(\mathfrak{L})$ be the class of schemata ϕ such that $\mathit{unfold}(\phi\{\mathtt{n} \leftarrow \mathtt{n}+1\})$ is $\mathfrak{L}$-dominated and such that the upper bound of all iterations in ϕ is $\mathtt{n}$.

Lemma 1. *Let $\mathfrak{L}$ be a stably decomposable frame. $\rho^* \circ \mathit{unfold}$ is an $\mathfrak{S}(\mathfrak{L})$-preserving simplification function*[2].

To ensure termination, we introduce a contraction operation: $\phi \wedge \phi \rightarrow \phi$ which is applied modulo the usual AC properties of the connective $\wedge$. Obviously this rule preserves equivalence. A set of schemata is *finite up to contraction* if its normal form by the previous rule is finite.

Theorem 3. *Let $\mathfrak{L}$ be a stably decomposable frame. If $\phi \in \mathfrak{S}(\mathfrak{L})$ then $\mathcal{E}_{\rho^* \circ \mathit{unfold}}(\phi)$ is finite up to contraction*[3]*. Thus the satisfiability problem is decidable for $\mathfrak{S}(\mathfrak{L})$.*

4 Examples of stably decomposable frames

Theorems 2 and 3 define a procedure for deciding the satisfiability of schemata in $\mathfrak{S}(\mathfrak{L})$. However, it relies on the fact that $\mathfrak{L}$ is stably decomposable, and on the existence of a function Δ satisfying the conditions of Definition 8. Thus, it would be of no use if no concrete example of (reasonably expressive) stably decomposable frame could be exhibited. The purpose of the present section is precisely to turn this abstract and generic result into concrete decision procedures.

4.1 Literals containing at most one index

The first example is independent of the theory $\mathcal{T}$. Intuitively, it corresponds to the case in which each literal contains at most one index. Let $\mathfrak{L}_\diamond$ be the set of flattened literals λ such that $\mathit{Ind}(\lambda) \in \{\{\mathtt{n}\}, \{\mathtt{n}+1\}, \{\mathtt{n}+2\}, \{\mathtt{i}\}, \{\mathtt{i}+1\}, \{0\}\}$. It is easy to check that $\mathfrak{L}_\diamond$ is a frame (it is finite if the signature is finite). Let $\Delta_\diamond$ be the function defined as follows:

$$\Delta_\diamond(\lambda_1 \wedge \cdots \wedge \lambda_k) \stackrel{\text{def}}{=} \begin{cases} \mathtt{true} & \text{if } (\lambda_1 \wedge \cdots \wedge \lambda_k)\{\mathtt{n} \leftarrow 0\} \text{ is } \mathcal{T}\text{-satisfiable} \\ \mathtt{false} & \text{otherwise.} \end{cases}$$

Theorem 4. *$\mathfrak{L}_\diamond$ is stably decomposable w.r.t. $\Delta_\diamond$.*

For instance, any purely propositional schema (i.e. any schema in which all atoms are non-equational) is in $\mathfrak{S}(\mathfrak{L}_\diamond)$[4]. Such schemata are essentially equivalent to the ones considered in [1]. The function $\Delta_\diamond$ should be compared with the pure literal rule in [1] that serves a similar purpose. The intuition is that the interpretation of the non-$\mathtt{n}$-elementary literals does not interfere with the one of $\mathtt{n}$-elementary literals. Notice that the analysis is much simpler in the present paper due to the strong syntactic restrictions.

[2] See Definition 4 for the notion of $\mathfrak{S}(\mathfrak{L})$-preserving function.
[3] See Definition 3 for the notation $\mathcal{E}_\Gamma(\phi)$.
[4] Provided indices greater than $\mathtt{n}+2$ or 0 are eliminated as explained in Section 2.3.

4.2 Ordered theories

The second example is more specific and also more complex. We assume that the signature contains a predicate symbol $\preceq$ interpreted as a non-strict ordering (in $\mathcal{T}$). Let $\mathcal{C}_{\approx}, \mathcal{C}_{\preceq}$ be two disjoint sets of indexed constant symbols. Intuitively, the constants in $\mathcal{C}_{\preceq}$ will only occur *at the root level* in non-strict inequations or equations, whereas the ones in $\mathcal{C}_{\approx}$ only occur in equations of some particular form. More precisely, we assume that every constant symbol $a \in \mathcal{C}_{\approx}$ is mapped to a finite set of terms $\theta(a)$, intended to denote the set of terms u such that $a_{\mathtt{n}+2} \approx u$ is allowed to occur in the considered schema. Furthermore, we assume that for all $u, v \in \theta(a)$, there exists an iteration-free $\mathtt{n}$-elementary schema $\tau(u \approx v)$ such that $\tau(u \approx v) \equiv_{\mathcal{T}} u \approx v$. The intuition is as follows. If u and v occur in $\theta(a)$, then the considered schema will possibly contain a conjunction of the form $a_{\mathtt{n}+2} \approx u \wedge a_{\mathtt{n}+2} \approx v$. As explained in Section 3.2, the symbol $a_{\mathtt{n}+2}$ will have to be eliminated (since it is non-$\mathtt{n}$-elementary) by applying an appropriate function Δ. But to this purpose, one necessarily has to ensure that the equation $u \approx v$ holds. The existence of the function τ guarantees that this property can be expressed as an $\mathtt{n}$-*elementary* schema.

Definition 9. *Let $\mathfrak{L}_{\preceq}$ be the set of literals λ satisfying one of the following conditions:*

- *λ is of the form $u \preceq v$[5], where each of the u, v is either a non-indexed term or of the form a_α where $a \in \mathcal{C}_{\preceq}$ and $\alpha \in \{\mathtt{n}, \mathtt{n}+1, \mathtt{n}+2, \mathtt{i}, \mathtt{i}+1\}$.*
- *λ is of the form $a_{\mathtt{n}+2} \approx u$ where $a \in \mathcal{C}_{\approx}$ and $u \in \theta(a)$.*
- *λ is of the form $a_{\mathtt{i}+1} \approx v$ (resp. $a_{\mathtt{n}+1} \approx v$) where $a \in \mathcal{C}_{\approx}$ and $v\{\mathtt{i} \leftarrow \mathtt{n}+1\} \in \theta(a)$ (resp. $v\{\mathtt{n} \leftarrow \mathtt{n}+1\} \in \theta(a)$).*

It is easy to check that $\mathfrak{L}_{\preceq}$ is a frame. We assume furthermore that for every $a \in \mathcal{C}_{\approx}$ and for all terms $u, v \in \theta(a)$, $\tau(u \approx v)$ is $\mathfrak{L}_{\preceq}$-dominated.

Before proceeding, we give a concrete example of a theory $\mathcal{T}$ for which $\theta(a)$ and τ can be defined (it will be used in forthcoming examples).

Example 3. Assume that `Sorts` contains in particular the sort symbols `nat`, `int` and `real` with their usual meanings. We assume that the signature contains the usual functions $+$ and $\preceq$[6] and built-in constant symbols $0, \ldots, k$ of sort `nat`. If $a : \mathtt{s} \in \mathcal{C}_{\approx}$, we define $\theta(a)$ as the set containing all terms in $0, \ldots, k$ (if $\mathtt{s}$ is `nat`) and all terms of the form $a_{\mathtt{n}+1} + u$ where u is either a non-indexed term or of the form $b_{\mathtt{n}+1}$ where $b \in \mathcal{C}_{\preceq}$. Then the function τ can be defined as follows:

- $\tau(a_{\mathtt{n}+1} + u \approx a_{\mathtt{n}+1} + v) \stackrel{\text{def}}{=} u \approx v$.
- $\tau(l \approx l') \stackrel{\text{def}}{=} l \approx l'$ if $Ind(l \approx l') = \emptyset$.
- $\tau(a_{\mathtt{n}+1} + u \approx l) \stackrel{\text{def}}{=} \bigvee_{l_1+l_2=l}(a_{\mathtt{n}+1} \approx l_1 \wedge u \approx l_2)$ if $l \in \{0, \ldots, k\}$. Note that the number of pairs (l_1, l_2) such that $l_1 + l_2 = l$ must be finite since by definition of $\theta(a)$, a (and thus l, l_1 and l_2) must be of sort `nat`. Hence this iteration is *not* a formal one but belongs to the meta-language. This would not be the case if a was of sort `int` or `real`.

[5] Of course, equations $u \approx v$ can also be considered, as abbreviations for $u \preceq v \wedge v \preceq u$.

[6] For readability, we use the same notation for the symbols $+$ and $\preceq$ whatever may be the type of their arguments.

It is easy to check that this function τ satisfies the desired properties.

Definition 10. *Let $\Delta_{\preceq}$ be the function defined as follows. For every conjunction of literals ϕ, we denote by $E(\phi)$ the smallest set of schemata such that:*

- *If ϕ contains two literals of the form $a_{\mathtt{n}+2} \approx u$ and $a_{\mathtt{n}+2} \approx v$ then $\tau(u \approx v) \in E(\phi)$.*
- *If $\phi \models u \preceq v$, $u \preceq v$ is an $\mathtt{n}$-elementary literal in $\mathfrak{L}_{\preceq}$ and $u \neq v$ then $u \preceq v \in E(\phi)$.*

We define: $\Delta_{\preceq}(\phi) \stackrel{def}{=} \bigwedge_{\psi \in E(\phi)} \psi$. Notice that $E(\phi)$ is necessarily finite.

Example 4. Let $a : \mathtt{nat}, b : \mathtt{int} \in \mathcal{C}_{\approx}$, $c : \mathtt{int}, d : \mathtt{int} \in \mathcal{C}_{\preceq}$, $e : \mathtt{int}$ and $f : \mathtt{int}$. Let $\theta(a) = \{a_{\mathtt{n}+1} + 1, 0, 1, 2\}$ and $\theta(b) = \{b_{\mathtt{n}+1} + c_{\mathtt{n}+1}, b_{\mathtt{n}+1} + e\}$.

Let ϕ be the conjunction of the following literals:

$$a_{\mathtt{n}+2} \approx a_{\mathtt{n}+1} + 1 \qquad a_{\mathtt{n}+2} \approx 2 \qquad b_{\mathtt{n}+2} \approx b_{\mathtt{n}+1} + c_{\mathtt{n}+1}$$
$$b_{\mathtt{n}+2} \approx b_{\mathtt{n}+1} + e \qquad c_{\mathtt{n}+1} \preceq d_{\mathtt{n}+2} \qquad d_{\mathtt{n}+2} \preceq d_{\mathtt{n}+1} \qquad d_{\mathtt{n}+2} \preceq f + 1$$

Then $\Delta_{\preceq}(\phi)$ is the conjunction of the following schemata:

$$a_{\mathtt{n}+1} \approx 1 \qquad c_{\mathtt{n}+1} \approx e \qquad c_{\mathtt{n}+1} \preceq d_{\mathtt{n}+1} \qquad c_{\mathtt{n}+1} \preceq f + 1$$

Theorem 5. *$\mathfrak{L}_{\preceq}$ is stably decomposable w.r.t. $\Delta_{\preceq}$.*

Another trivial example of stably decomposable sets of literals that we do not develop here, is the one in which every constant symbol indexed by an expression $\mathtt{n}+l$ where $l > 1$, is of a finite sort. Indeed, in this case all such constants can be straightforwardly eliminated by replacing them by each possible value (yielding a disjunction of $\mathtt{n}$-elementary schemata).

5 Examples

We provide in this section some examples of application of our technique.

Example 5. Let ϕ be the schema considered in the Introduction: $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}} (a_{\mathtt{i}+1} \succeq a_{\mathtt{i}}) \wedge \bigwedge_{\mathtt{i}=0}^{\mathtt{n}} (b_{\mathtt{i}+1} \preceq b_{\mathtt{i}}) \wedge a_0 \succeq b_0 \wedge a_{\mathtt{n}+1} \preceq c \wedge b_{\mathtt{n}+1} \succeq c + 1$.

We compute the set of schemata $\mathcal{E}_{\rho^* \circ unfold}(\phi)$. According to the definition, $\mathtt{n}$ must be instantiated by $\mathtt{n}+1$ and the iterations are unfolded, yielding: $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}} (a_{\mathtt{i}+1} \succeq a_{\mathtt{i}}) \wedge a_{\mathtt{n}+2} \succeq a_{\mathtt{n}+1} \wedge \bigwedge_{\mathtt{i}=0}^{\mathtt{n}} (b_{\mathtt{i}+1} \preceq b_{\mathtt{i}}) \wedge b_{\mathtt{n}+2} \preceq b_{\mathtt{n}+1} \wedge a_0 \succeq b_0 \wedge a_{\mathtt{n}+2} \preceq c \wedge b_{\mathtt{n}+2} \succeq c + 1$. In order to get rid of the symbols indexed by $\mathtt{n}+2$, we apply the rules in ρ. Since the schema is already a conjunction of iterations and literals, no rule applies, except Elimination. The conjunction of literals that are not $\mathtt{n}$-elementary is $a_{\mathtt{n}+2} \succeq a_{\mathtt{n}+1} \wedge b_{\mathtt{n}+2} \preceq b_{\mathtt{n}+1} \wedge a_{\mathtt{n}+2} \preceq c \wedge b_{\mathtt{n}+2} \succeq c + 1$. Applying the function $\Delta_{\preceq}$ (see Definition 10), we obtain: $c \succeq a_{\mathtt{n}+1} \wedge c + 1 \preceq b_{\mathtt{n}+1}$. Replacing the previous conjunction by its image by $\Delta_{\preceq}$ yields a schema that is actually identical to the first one. Hence the procedure stops (no further schema is generated) and we get $\mathcal{E}_{\rho^* \circ unfold}(\phi) = \{\phi\}$. By Theorem 2, the $\mathcal{T}$-satisfiability of ϕ is thus equivalent to the one of $\phi\{\mathtt{n} \leftarrow 0\}$ which can be easily tested by any SMT-solver.

Example 6. Consider the algorithm below, counting the number of occurrences o of an element e in an array t. We want to check that if the final value of o is 1 then the formula $\forall i, j,\ a_{\mathtt{i}} \approx e \wedge a_j \approx e \Rightarrow i \approx j$ holds. This is modeled by a schema ϕ defined as follows ($o_{\mathtt{i}} : \mathtt{nat}$ denotes the value of o at step i and $t_{\mathtt{i}} : \mathtt{int}$ is $t[\mathtt{i}]$, notice that we cannot use the theory of arrays, since no stably decomposable frame has been defined for this theory – this is left to future work).

```
i ← 0
o ← 0
while i ⪯ n do
  if t[i] = e then
    o ← o + 1
  end if
  i ← i + 1
end while
```

$$
\begin{array}{c}
\phi: \\
\\
o_0 \approx 0 \\
\bigwedge_{i=0}^{n} (t_i \approx e \Rightarrow o_{i+1} \approx o_i + 1) \\
\bigwedge_{i=0}^{n} (t_i \not\approx e \Rightarrow o_{i+1} \approx o_i) \\
o_{n+1} \approx 1 \\
\bigvee_{i=0}^{n} (i \approx m \wedge t_i \approx e) \\
\bigvee_{i=0}^{n} (i \approx k \wedge t_i \approx e) \\
m \not\approx k
\end{array}
$$

m, k are additional parameters interpreted as elements of $[0, n]$. These parameters and the literals $i \approx m$, $i \approx k$ and $m \not\approx k$ can be encoded in our language as explained in Section 2.3 (we omit the translation for readability). t is in $\mathcal{C}_{\preceq}$ and o is in $\mathcal{C}_{\approx}$. The schema is in $\mathfrak{S}(\mathfrak{L}_{\preceq})$. The reader can check that $\mathcal{E}_{\rho^* \circ unfold}(\phi) = \{\phi, \psi_1, \psi_2, \psi_3\}$, where ψ_1, ψ_2 and ψ_3 are defined respectively by:

$$
\begin{array}{c|c|c}
\psi_1: & \psi_2: & \psi_3: \\
o_0 \approx 0 & o_0 \approx 0 & o_0 \approx 0 \\
\bigwedge_{i=0}^{n} t_i \approx e \Rightarrow o_{i+1} \approx o_i + 1 & \bigwedge_{i=0}^{n} t_i \approx e \Rightarrow o_{i+1} \approx o_i + 1 & \bigwedge_{i=0}^{n} t_i \approx e \Rightarrow o_{i+1} \approx o_i + 1 \\
\bigwedge_{i=0}^{n} t_i \not\approx e \Rightarrow o_{i+1} \approx o_i & \bigwedge_{i=0}^{n} t_i \not\approx e \Rightarrow o_{i+1} \approx o_i & \bigwedge_{i=0}^{n} t_i \not\approx e \Rightarrow o_{i+1} \approx o_i \\
o_{n+1} \approx 0 & o_{n+1} \approx 0 & o_{n+1} \approx 0 \\
t_{n+1} \approx e & t_{n+1} \approx e & t_{n+1} \approx e \\
n + 1 \approx m & n + 1 \approx k & \bigvee_{i=0}^{n} i \approx m \wedge t_i \approx e \\
\bigvee_{i=0}^{n} i \approx k \wedge t_i \approx e & \bigvee_{i=0}^{n} i \approx m \wedge t_i \approx e & \bigvee_{i=0}^{n} i \approx k \wedge t_i \approx e \\
m \not\approx k & m \not\approx k & m \not\approx k
\end{array}
$$

In order to check that ϕ is $\mathcal{T}$-unsatisfiable, one only has to test the $\mathcal{T}$-satisfiability of the sentences $\phi\{n \leftarrow 0\}$, $\psi_1\{n \leftarrow 0\}$, $\psi_2\{n \leftarrow 0\}$ and $\psi_3\{n \leftarrow 0\}$.

The next example is slightly more complex, hence we only show the encoding (to give a taste of the expressive power of the class $\mathfrak{S}(\mathfrak{L}_{\preceq})$).

Example 7. Consider the algorithm to the right, inserting a new element in a sorted sequence. We want to check that the obtained sequence a' is sorted, which is modeled by the schema on the left.

$b_0 \wedge \neg b_{n+1} \wedge (\bigwedge_{i=0}^{n} \phi) \wedge \psi$, where:

- b is true inside the first loop, false otherwise.
- ϕ is defined as follows: $(\neg b_i \vee \text{new} + 1 \preceq a_i \vee a'_i \approx a_i \wedge b_{i+1}) \wedge (\neg b_i \vee a_i \preceq \text{new} \vee a'_i \approx \text{new} \wedge \neg b_{i+1}) \wedge (b_{i+1} \vee a'_{i+1} \approx a_i) \wedge (b_i \vee \neg b_{i+1})$.
- ψ states the fact that a' is not sorted: $\bigvee_{i=0}^{n} (a'_i \succ c \wedge a'_{i+1} \preceq c)$.

It can be checked that the obtained schema is in $\mathfrak{S}(\mathfrak{L}_{\preceq})$. a, a' both occur in $\mathcal{C}_{\preceq}$.

```
i ← 0
while a_i ⪯ new ∧ i ⪯ n do
  a'_i ← a_i
  i ← i + 1
end while
a'_i ← new
while i ⪯ n do
  a'_{i+1} ← a_i
  i ← i + 1
end while
```

6 Conclusion

A logic has been defined for reasoning on parameterized families of SMT-problems and a sound and complete (w.r.t. satisfiability) proof procedure has been designed. It does not terminate in general (the logic is proven to be undecidable) but we have devised semantic conditions on the underlying theory and

on the considered class of formulae that ensure that this proof procedure can be turned into a decision procedure by adding appropriate simplification rules. Then, concrete examples of theories and classes of schemata satisfying these conditions have been provided. Some simple examples of application have also been proposed. Our method relies on the use of an external decision procedure for the underlying theory. It applies to a wide range of theories (provided they are decidable). In the present work, we mainly focus on examples in verification, but one could also handle for instance schemata of formulae in (decidable) modal or description logics.

The implementation of this technique is part of future work. Another obvious line of research is to identify other classes of stably decomposable frames (see Section 3.2) in order to extend the scope of our results (in particular, the important theory of arrays should be considered). Concerning potential applications in verification, automatic procedures for extracting schemata modeling the algorithms as the ones in Section 5 ought to be devised and comparison with the numerous existing techniques should be provided. A longer term goal would be to consider quantification, either as standard quantification such as $\forall x, \forall y,\ p(x, y)$ or of *schemata of quantifications* such as $\forall x_1, \ldots, x_n,\ p(x_1, \ldots, x_n)$ (where the indexed variables and dots are *part of the language*).

References

1. V. Aravantinos, R. Caferra, and N. Peltier. A schemata calculus for propositional logic. In *TABLEAUX 09 (International Conference on Automated Reasoning with Analytic Tableaux and Related Methods)*, volume 5607 of *LNCS*, pages 32–46. Springer, 2009.
2. V. Aravantinos, R. Caferra, and N. Peltier. Decidability and undecidability results for propositional schemata. *Journal of Artificial Intelligence Research*, 2011. Accepted, to appear.
3. C. Barrett, R. Sebastiani, S. Seshia, and C. Tinelli. Satisfiability modulo theories. In A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 26, pages 825–885. IOS Press, 2009.
4. A. R. Bradley, Z. Manna, and H. B. Sipma. What's decidable about arrays? In E. A. Emerson and K. S. Namjoshi, editors, *Proc. VMCAI-7*, volume 3855 of *LNCS*, pages 427–442. Springer, 2006.
5. L. M. de Moura and N. Bjorner. Efficient E-Matching for SMT Solvers. In F. Pfenning, editor, *21st International Conference on Automated Deduction*, volume 4603 of *LNCS*, pages 183–198. Springer, 2007.
6. D. Dutertre and L. de Moura. The YICES SMT-solver, 2006. In SMT-COMP: Satisfiability Modulo Theories Competition. Available at http://yices.csl.sri.com.
7. Y. Ge and L. M. de Moura. Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In A. Bouajjani and O. Maler, editors, *CAV 2009*, volume 5643 of *LNCS*, pages 306–320. Springer, 2009.
8. P. Habermehl, R. Iosif, and T. Vojnar. What else is decidable about integer arrays? In R. M. Amadio, editor, *FoSSaCS*, volume 4962 of *Lecture Notes in Computer Science*, pages 474–489. Springer, 2008.

A Proof of Proposition 1

The only iterations occurring in ϕ are of the form $\bigvee_{\mathtt{i}=0}^{k} \psi$ (resp. $\bigwedge_{\mathtt{i}=0}^{k} \psi$) where $k \in \mathbb{N}$. Such an iteration[7] is equivalent to (hence can be replaced by): $\psi\{\mathtt{i} \leftarrow 0\} \vee \psi\{\mathtt{i} \leftarrow 1\} \vee \cdots \vee \psi\{\mathtt{i} \leftarrow k\}$ (resp. $\psi\{\mathtt{i} \leftarrow 0\} \wedge \psi\{\mathtt{i} \leftarrow 1\} \wedge \cdots \vee \psi\{\mathtt{i} \leftarrow k\}$). After all iterations have been replaced, one obtains a schema containing no iteration and no occurrence of $\mathtt{n}$, i.e. a sentence. By hypothesis, the $\mathcal{T}$-satisfiability problem is decidable for sentences.

B Proof of Theorem 1

In this section, we assume that $\mathcal{T}$ simply contains all the interpretations on the considered signature (i.e. there are no built-in symbols). Notice that the signature contains indexed constant symbols and non-indexed function symbols (see below). The proof is by reduction to the Post correspondence problem. Let $k \in \mathbb{N}$ and let $(\Gamma^1, \ldots, \Gamma^k)$, $(\Lambda^1, \ldots, \Lambda^k)$ be two sequences of words over an alphabet $\mathcal{A}$. For every word $w \in \mathcal{A}^*$, $|w|$ denotes the length of w (i.e. the number of characters). $w \cdot w'$ denotes the concatenation of the words w and w'. If $\Omega \in \{\Gamma, \Lambda\}$, $i \in [1, k]$ and $j \in [1, |\Omega^i|]$, we denote by $\Omega^i(j)$ the j-th character of the word Ω^i (we do not use indices to avoid confusion with indexed symbols in the language). We recall that the aim of the Post problem is to determine whether there exists a non-empty sequence of indices $(\delta_1, \ldots, \delta_l)$ such that $\Gamma^{\delta_1} \cdot \ldots \cdot \Gamma^{\delta_l} = \Lambda^{\delta_1} \cdot \ldots \cdot \Lambda^{\delta_l}$. It is well known that this problem is not decidable. The sequence $\delta_1, \ldots, \delta_l$ is the *sequence solution* and $\Gamma^{\delta_1} \cdot \ldots \cdot \Gamma^{\delta_l}$ (or equivalently $\Lambda^{\delta_1} \cdot \ldots \cdot \Lambda^{\delta_l}$) is the *word solution*.

Signature

We consider three different sorts $\mathtt{A}$ (intended to be interpreted as elements of $\mathcal{A}$), $\mathtt{ind}$ (elements of $[1, k]$) and $\mathtt{seq}$ (sequences of elements of $[1, k]$). We assume that all the symbols $\spadesuit$ in $\mathcal{A}$ are mapped to pairwise distinct non-indexed constant symbols of sort $\mathtt{A}$. For the sake of readability, the image of $\spadesuit$ is also denoted by $\spadesuit$. Similarly, each natural number in $[1, k]$ is considered as a non-indexed constant symbol of sort $\mathtt{ind}$. We encode sequences of indices (i.e. elements of $[1, k]^*$) using two non-built-in function symbols $head : \mathtt{seq} \rightarrow \mathtt{ind}$ and $tail : \mathtt{seq} \rightarrow \mathtt{seq}$, which return respectively the first element of the sequence and its tail. The constant symbol $d : \mathtt{seq}$ denotes the sequence solution $\delta = (\delta_1, \ldots, \delta_l)$ and $nil : \mathtt{seq}$ denotes the empty sequence.

We use two indexed constants $sol^\Omega : \mathtt{A}$ and $F^\Omega : \mathtt{seq}$ for each $\Omega \in \{\Gamma, \Lambda\}$. sol^Ω is used to stored the word solution $\Omega^{\delta_1} \cdot \ldots \cdot \Omega^{\delta_l}$, more precisely $sol^\Omega_{\mathtt{i}}$ is the $i+1$-th character of the word $\Omega^{\delta_1} \cdot \ldots \cdot \Omega^{\delta_l}$ (if $i \geq |\Omega^{\delta_1}| + \cdots + |\Omega^{\delta_l}|$ then the value of $sol^\Omega_{\mathtt{i}}$ is irrelevant). F^Ω contains the suffixes of the sequence solution corresponding to a given position in the word solution. More precisely, if i is of the form $|\Omega^{\delta_1}| + \cdots + |\Omega^{\delta_m}|$ (for some $m \in [0, l]$) then $F^\Omega_{\mathtt{i}}$ contains $(\delta_{m+1}, \delta_{m+2}, \ldots, \delta_l)$. If i is not of this form then the value of F^Ω is equal to a special constant symbol $\perp$ of sort $\mathtt{seq}$. In particular, if $i = 0$ then $F^\Omega_{\mathtt{i}}$ contains the whole sequence (corresponding to the case $m = 0$).

[7] Notice that such an iteration is never empty because $k \geq 0$.

Example 8. Let $\mathcal{A} = \{\alpha, \beta, \gamma, \pi\}$. Let $\Gamma = \{\alpha\beta, \alpha\gamma, \pi\}$. In this example, we have $k = 3$. Assume that the sequence solution δ is $\{1, 3, 2\}$. The corresponding word solution is $\alpha\beta\pi\alpha\gamma$. The following array specifies, for each index i, the corresponding values of sol^{Γ} and F^{Γ}.

i	0	1	2	3	4	5
sol	α	β	π	α	γ	$\bullet$
F	$(1,3,2)$	$\bot$	$(3,2)$	(2)	$\bot$	$()$

Encoding

The problem is specified by the following axioms (parameterized by k, Γ and Λ). Notice that, for the sake of readability, arbitrary translations are used in the indices (see for instance Axiom 4, index $i+m$). As explained in Section 2.3, they can be easily encoded in the language. The parameter $\mathtt{n}$ encodes the length of the word solution.

(1): $F_0^{\Omega} \approx d$
for each $\Omega \in \{\Gamma, \Lambda\}$
% The initial sequence of each word is the solution sequence.
(2): $d \not\approx nil \wedge d \not\approx \bot$
% The solution sequence is not empty and distinct from $\bot$.
(3): $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}} \left(F_{\mathtt{i}}^{\Omega} \approx \bot \vee F_{\mathtt{i}}^{\Omega} \approx nil \vee head(F_{\mathtt{i}}^{\Omega}) \approx 1 \vee \cdots \vee head(F_{\mathtt{i}}^{\Omega}) \approx k \right)$
for each $\Omega \in \{\Gamma, \Lambda\}$
% The first element of each (non-empty) suffix is in $[1, k]$.
(4): $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}} \left(F_{\mathtt{i}}^{\Omega} \approx \bot \vee head(F_{\mathtt{i}}^{\Omega}) \not\approx l \vee sol_{\mathtt{i}+m}^{\Omega} \approx \Omega^l(m+1) \right)$
for each $\Omega \in \{\Gamma, \Lambda\}$, $l \in [1, k]$ and $m \in [0, |\Omega^l|[$
% Relate the value of $sol_{\mathtt{i}}^{\Omega}, sol_{\mathtt{i}+1}^{\Omega}, \ldots$ to the value of $F_{\mathtt{i}}^{\Omega}$
(5): $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}} \left(F_{\mathtt{i}}^{\Omega} \approx \bot \vee head(F_{\mathtt{i}}^{\Omega}) \not\approx l \vee F_{\mathtt{i}+m}^{\Omega} \approx \bot \right)$
for each $\Omega \in \{\Gamma, \Lambda\}$, $l \in [1, k]$ and $m \in [0, |\Omega^l|[$
% Store $\bot$ at the indices not corresponding to a sequence.
(6): $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}} \left(F_{\mathtt{i}}^{\Omega} \approx \bot \vee head(F_{\mathtt{i}}^{\Omega}) \not\approx l \vee F_{\mathtt{i}+|\Omega^l|}^{\Omega} \not\approx \bot \wedge F_{\mathtt{i}+|\Omega^l|}^{\Omega} \approx tail(F_{\mathtt{i}}^{\Omega}) \right)$
for each $\Omega \in \{\Gamma, \Lambda\}$ and for each $l \in [1, k]$
% The next value of F^{Ω} is equal to the tail of the previous sequence.
(7) $\bigwedge_{\mathtt{i}=0}^{\mathtt{n}} sol_{\mathtt{i}}^{\Gamma} \approx sol_{\mathtt{i}}^{\Lambda}$
% The word indices corresponding to each sequence are identical.
(8) $F_{\mathtt{n}+1}^{\Omega} \approx nil$
for each $\Omega \in \{\Gamma, \Lambda\}$
% Both sequences end at $\mathtt{n}$.

We denote by $\phi(\Gamma, \Lambda)$ the conjunction of Axioms 1-8. Notice that the obtained set of axioms is finite (if the sequences Γ and Λ are fixed for a given instance of the Post correspondence problem).

Lemma 2. *$\phi(\Gamma, \Lambda)$ is satisfiable iff there exists a non-empty sequence of indices $\delta_1, \ldots, \delta_l$ such that $\Gamma^{\delta_1}.\ldots.\Gamma^{\delta_l} = \Lambda^{\delta_1}.\ldots.\Lambda^{\delta_l}$.*

Proof. This is easy to check from the previous explanations. □

C Proof of Theorem 2

Proposition 4. *Let ϕ be a schema and let I be an interpretation. $\langle\phi\rangle^{I\circ\{\mathtt{n}\mapsto\alpha\}} = \langle\phi\{\mathtt{n}\leftarrow\alpha\}\rangle^{I}$.*

Proof. By a straightforward induction on ϕ. □

Lemma 3. *Let Γ be a simplification function. For every schema ϕ, for every $\psi \in \mathcal{E}_\Gamma(\phi)$ and for every interpretation $I \in \mathcal{T}$ validating ψ, there exists a schema $\psi' \in \mathcal{E}_\Gamma(\phi)$ such that $I \models \psi'\{\mathtt{n}\leftarrow 0\}$.*

Proof. The proof is by induction on $\langle\mathtt{n}\rangle^I$. Assume that $\langle\mathtt{n}\rangle^I = 0$. Then $I \models \psi\{\mathtt{n}\leftarrow 0\}$. Since $\psi \in \mathcal{E}_\Gamma(\phi)$, the desired result trivially holds for $\psi' = \psi$. Now, assume that $\langle\mathtt{n}\rangle^I > 0$. By Property 2 in Definition 3, $\Gamma(\psi\{\mathtt{n}\leftarrow \mathtt{n}{+}1\}) \subseteq \mathcal{E}_\Gamma(\phi)$. Let $J = I\circ\{\mathtt{n}\mapsto \langle\mathtt{n}\rangle^I{-}1\}$. We have $\langle\psi\{\mathtt{n}\leftarrow \mathtt{n}+1\}\rangle^J = \langle\psi\{\mathtt{n}\leftarrow \mathtt{n}+1\}\rangle^{I\circ\{\mathtt{n}\mapsto\langle\mathtt{n}\rangle^I-1\}}$. By Proposition 4, this is equal to $\langle\psi\rangle^I$ i.e. to $\mathtt{true}$ by hypothesis. So $J \models \psi\{\mathtt{n}\leftarrow \mathtt{n}{+}1\}$. By Point 1 in Definition 1, we deduce that $J \models \psi'$ for some $\psi' \in \Gamma(\psi\{\mathtt{n}\leftarrow \mathtt{n}+1\})$. Thus by the induction hypothesis (since $\langle\mathtt{n}\rangle^J < \langle\mathtt{n}\rangle^I$) there exists a schema $\psi'' \in \mathcal{E}_\Gamma(\phi)$ such that $J \models \psi''\{\mathtt{n}\leftarrow 0\}$. Since $\psi''\{\mathtt{n}\leftarrow 0\}$ contains no occurrence of $\mathtt{n}$, I and J coincide on $\psi''\{\mathtt{n}\leftarrow 0\}$, hence $I \models \psi''\{\mathtt{n}\leftarrow 0\}$. □

Assume that ϕ is $\mathcal{T}$-satisfiable. Let $I \in \mathcal{T}$ be an interpretation satisfying ϕ. By Point 1 in Definition 3, $\phi \in \mathcal{E}_\Gamma(\phi)$. By Lemma 3, there exists $\psi \in \mathcal{E}_\Gamma(\phi)$ such that $I \models \psi\{\mathtt{n}\leftarrow 0\}$.

Now, assume that $\mathcal{E}_\Gamma(\phi)$ contains a schema ψ such that $\psi\{\mathtt{n}\leftarrow k\}$ is $\mathcal{T}$-satisfiable (for some *arbitrary* natural number k, in particular if $k = 0$). This means that ψ has a model $I \in \mathcal{T}$ such that $\langle\mathtt{n}\rangle^I = k$. We prove, by induction on the construction of the set $\mathcal{E}_\Gamma(\phi)$ that ϕ is $\mathcal{T}$-satisfiable. If $\psi = \phi$ then the proof is straightforward. If $\psi \in \Gamma(\psi'\{\mathtt{n}\leftarrow \mathtt{n}+1\})$ for some $\psi' \in \mathcal{E}_\Gamma(\phi)$, then by Point 2 in Definition 1, there exists an interpretation $J \in \mathcal{T}$ such that $J \models \psi'\{\mathtt{n}\leftarrow \mathtt{n}+1\}$ and $\langle\mathtt{n}\rangle^J = \langle\mathtt{n}\rangle^I = k$. Then by Proposition 4, the interpretation $J' = J\circ\{\mathtt{n}\mapsto\langle\mathtt{n}\rangle^I+1\}$ must be a model of ψ'. Since $\langle\mathtt{n}\rangle^J = k$, $\psi'\{\mathtt{n}\leftarrow k{+}1\}$ is $\mathcal{T}$-satisfiable and by the induction hypothesis this implies that ϕ is $\mathcal{T}$-satisfiable.

D Proof of Proposition 3

The Closure and $\vee$-Elimination rules strictly decrease the number of logical symbols occurring in the branches hence termination is obvious. The rule Elimination applies at most once on every branch. Indeed, by definition, $\Delta(\lambda_1 \wedge \cdots \wedge \lambda_k)$ only contains $\mathtt{n}$-elementary literals. Thus by the application of the rule (and because the Elimination rule is defined to apply on the set of *all* non-$\mathtt{n}$-elementary literals in the branch), every literal in the obtained branch must be $\mathtt{n}$-elementary. Consequently, the branch obtained after applying Elimination is necessarily irreducible.

E Proof of Lemma 1

Proposition 5. *For every schema ϕ, unfold(ϕ) $\equiv_{\mathcal{T}} \phi$.*

Proof. The replacement obviously preserves equivalence. □

Proposition 6. *If $\phi \rightsquigarrow \Psi \cup \{\psi\}$ and $\psi \rightsquigarrow \Psi'$ then $\phi \rightsquigarrow \Psi \cup \Psi'$.*

Proof. Immediate. □

Lemma 4. *Let $\mathfrak{L}$ be a frame. Let I and J be two interpretations such that $I \sim^{\mathfrak{L}} J$. For all $\mathfrak{L}$-dominated $\mathtt{n}$-elementary schemata ϕ irreducible by unfold we have:*

$$I \models \phi \text{ iff } J \models \phi.$$

Proof. It suffices to prove that for every $k \in [0, \langle \mathtt{n} \rangle^I]$ we have $\langle \phi\{\mathtt{i} \leftarrow k\}\rangle^I = \langle \phi\{\mathtt{i} \leftarrow k\}\rangle^J$. The proof is by induction on ϕ.

- If ϕ is a literal, then since ϕ is $\mathfrak{L}$-dominated, ϕ necessarily occurs in $\mathfrak{L}$. Since $k \in [0, \langle \mathtt{n} \rangle^I]$, we have by the definition of $\sim^{\mathfrak{L}}$: $\langle \phi\{\mathtt{i} \leftarrow k\}\rangle^I = \langle \phi\{\mathtt{i} \leftarrow k\}\rangle^J$.
- If ϕ is of the form $\psi_1 \star \psi_2$ (for $\star \in \{\vee, \wedge\}$) then by the induction hypothesis we have $I \models \psi_i$ iff $J \models \psi_i$ (for $i = 1, 2$) thus $I \models \phi$ iff $J \models \phi$.
- Assume that ϕ is $\bigvee_{\mathtt{i}=0}^{\mathtt{n}} \psi$ (notice that by irreducibility w.r.t. *unfold*, the upper bound of the iterations must be $\mathtt{n}$). $\langle \phi \rangle^I = \mathtt{true}$ iff there exists $l \in [0, \langle \mathtt{n} \rangle^I]$ such that $\langle \psi\{\mathtt{i} \leftarrow l\}\rangle^I = \mathtt{true}$ i.e. by Proposition 4, $I \circ \{\mathtt{n} \mapsto l\} \models \psi$. Since $l \leq \langle \mathtt{n} \rangle^I$, by the induction hypothesis, $\langle \psi\{\mathtt{i} \leftarrow l\}\rangle^I = \langle \psi\{\mathtt{i} \leftarrow l\}\rangle^J$. Thus $\langle \psi \rangle^I = \langle \psi \rangle^J$.
- The proof is similar if ψ is an iterated conjunction.

□

We first prove that $\rho^* \circ \mathit{unfold}$ is a simplification function. Let $\psi \in \mathfrak{S}(\mathfrak{L})$. Let $\phi = \mathit{unfold}(\psi)$. We have to prove that $\psi \rightsquigarrow \rho^*(\phi)$. By Proposition 5 we have $\phi \equiv_{\mathcal{T}} \psi$ thus $\psi \rightsquigarrow \{\phi\}$. We prove that $\phi \rightsquigarrow \rho^*(\phi)$ (then the proof immediately follows from Proposition 6). By definition, $\rho^*(\phi)$ is an irreducible tableau constructed by the expansion rules in ρ from $\{\phi\}$. The proof is by induction on the number of rules applied to get $\rho^*(\phi)$. If no rule is applied then $\rho^*(\phi) = \{\phi\}$ and the proof is obvious. Otherwise, we distinguish several cases, according to the first rule applied on ϕ.

- If this rule is the Closure rule, then $\rho^*(\phi) = \emptyset$. In this case ϕ contains two contradictory literals, hence is unsatisfiable. Thus $\phi \rightsquigarrow \rho^*(\phi)$.
- If this rule is the $\vee$-Elimination rule, then ϕ is of the form $(\phi_1 \vee \phi_2) \wedge \psi$. The two obtained branches are $\phi_1 \wedge \psi$ and $\phi_2 \wedge \psi$. Thus $\rho^*(\phi) = \rho^*(\phi_1 \wedge \psi) \cup \rho^*(\phi_2 \wedge \psi)$. ϕ is equivalent to $(\phi_1 \wedge \psi) \vee (\phi_2 \wedge \psi)$ hence $\phi \rightsquigarrow \{\phi_1 \wedge \psi, \phi_2 \wedge \psi\}$. By the induction hypothesis, we have $\phi_1 \wedge \psi \rightsquigarrow \rho^*(\phi_1 \wedge \psi)$ and $\phi_2 \wedge \psi \rightsquigarrow \rho^*(\phi_2 \wedge \psi)$. By Proposition 6, we deduce that $\phi \rightsquigarrow \rho^*(\phi_1 \wedge \psi) \cup \rho^*(\phi_2 \wedge \psi) = \rho^*(\phi)$.

– If this rule is the Elimination rule, then ϕ is of the form $\lambda_1 \wedge \cdots \wedge \lambda_k \wedge \phi'$ where ϕ' contains no non-$\mathtt{n}$-elementary literals and $\lambda_1, \ldots, \lambda_k$ are non-$\mathtt{n}$-elementary literals. Moreover, the obtained branch is $\Delta(\lambda_1 \wedge \cdots \wedge \lambda_k) \wedge \phi'$. By definition, this branch must be irreducible, since $\Delta(\lambda_1 \wedge \cdots \wedge \lambda_k)$ contains only $\mathtt{n}$-elementary literals. Hence $\rho^*(\phi) = \{\Delta(\lambda_1 \wedge \cdots \wedge \lambda_k) \wedge \phi'\}$. Let $I \in \mathcal{T}$ be an interpretation satisfying ϕ. Since $\mathfrak{L}$ is stably decomposable w.r.t. Δ, we have $I \models \Delta(\lambda_1 \wedge \cdots \wedge \lambda_k)$. Thus $I \models \Delta(\lambda_1 \wedge \cdots \wedge \lambda_k) \wedge \phi'$. Conversely, let $I \in \mathcal{T}$ be an interpretation satisfying $\Delta(\lambda_1 \wedge \cdots \wedge \lambda_k) \wedge \phi'$. By definition of Δ, there exists an interpretation $J \sim^{\mathfrak{L}} I$ such that $J \models \lambda_1 \wedge \cdots \wedge \lambda_k$. By definition ϕ' contains no non-$\mathtt{n}$-elementary literals and is irreducible by *unfold*. By Lemma 4, we have $J \models \phi'$. Thus $J \models \phi$. Moreover, $\langle \mathtt{n} \rangle^I = \langle \mathtt{n} \rangle^J$, by definition of $\sim^{\mathfrak{L}}$.

Then we have to prove that $\rho^* \circ \mathit{unfold}$ is $\mathfrak{S}(\mathfrak{L})$-preserving. Let $\phi \in \mathfrak{S}(\mathfrak{L})$. Let $\psi \in \rho^*(\mathit{unfold}(\phi\{\mathtt{n} \leftarrow \mathtt{n}+1\}))$. We have to show that $\psi \in \mathfrak{S}(\mathfrak{L})$, i.e. that for every literal λ occurring in $\mathit{unfold}(\psi\{\mathtt{n} \leftarrow \mathtt{n}+1\})$, we have $\lambda \in \mathfrak{L}$. Let λ be a literal in $\mathit{unfold}(\psi\{\mathtt{n} \leftarrow \mathtt{n}+1\})$. By definition of *unfold*, there exists a literal λ' occurring in ψ such that either $\lambda = \lambda'\{\mathtt{n} \leftarrow \mathtt{n}+1\}$ or $\lambda = \lambda'\{\mathtt{i} \leftarrow \mathtt{n}+1\}$. We distinguish two cases:

– If λ' has been introduced by an application of the rule Elimination, then by definition $\lambda' \in \mathfrak{L}$. Moreover it is $\mathtt{n}$-elementary and ground. Thus we must have $\lambda = \lambda'\{\mathtt{n} \leftarrow \mathtt{n}+1\}$ (since λ' contains no occurrence of $\mathtt{i}$). By Condition 2 in Definition 6 (definition of a frame), we deduce $\lambda \in \mathfrak{L}$.
– Otherwise, λ' must occur in $\mathit{unfold}(\phi\{\mathtt{n} \leftarrow \mathtt{n}+1\})$. Furthermore, it must be $\mathtt{n}$-elementary. Since $\phi \in \mathfrak{S}(\mathfrak{L})$, $\lambda' \in \mathfrak{L}$. Thus $\lambda'\{\mathtt{n} \leftarrow \mathtt{n}+1\} \in \mathfrak{L}$, by Condition 2 in Definition 6. If $\lambda = \lambda'\{\mathtt{n} \leftarrow \mathtt{n}+1\}$ then the proof is completed. Otherwise, by Condition 1 in Definition 6 we have $\lambda'\{\mathtt{i} \leftarrow \mathtt{n}+1\} \in \mathfrak{L}$, i.e. $\lambda \in \mathfrak{L}$.

F Proof of Theorem 3

The *depth* of a schema or an iteration body is defined as usual: $\mathrm{depth}(\phi) \stackrel{\text{def}}{=} 0$ if ϕ is a literal, $\mathrm{depth}(\phi \star \psi) \stackrel{\text{def}}{=} \max(\mathrm{depth}(\phi), \mathrm{depth}(\psi)) + 1$ (if $\star = \vee, \wedge$) $\mathrm{depth}(\neg\phi) \stackrel{\text{def}}{=} \mathrm{depth}(\phi + 1)$, $\mathrm{depth}(\bigvee_{\mathtt{i}=0}^{\alpha+k} \phi) \stackrel{\text{def}}{=} \mathrm{depth}(\bigwedge_{\mathtt{i}=0}^{\alpha+k} \phi) \stackrel{\text{def}}{=} \mathrm{depth}(\phi) + 1$. Then we define the function $\mathrm{depth}_\wedge(\phi)$ as follows (it is identical to $\mathrm{depth}(\phi)$, except that all the conjunctions at root level are ignored): $\mathrm{depth}_\wedge(\phi) \stackrel{\text{def}}{=} \mathrm{depth}(\phi)$ if ϕ is not a conjunction, and $\mathrm{depth}_\wedge(\phi_1 \wedge \phi_2) \stackrel{\text{def}}{=} \max(\mathrm{depth}_\wedge(\phi_1), \mathrm{depth}_\wedge(\phi_2))$. We have the following:

Proposition 7. *Let $\mathfrak{L}$ be a frame. Let $d \in \mathbb{N}$. The set of schemata ϕ such that $\phi \in \mathfrak{S}(\mathfrak{L})$ and $\mathrm{depth}_\wedge(\phi) \leq d$ is finite up to contraction.*

Proof. By definition, ϕ can be written as $\phi_1 \wedge \cdots \wedge \phi_k$ where $\phi_1, \ldots, \phi_k$ are not conjunctions. Then we have $\mathrm{depth}_\wedge(\phi) = \max_{l \in [1,k]} \mathrm{depth}(\phi_l)$, hence $\forall l \in [1,k], \mathrm{depth}(\phi_l) \leq d$. Since the set of literals possibly occurring in $\phi_1, \ldots, \phi_k$ is finite and since the depth of $\phi_1, \ldots, \phi_k$ is bounded, the set of possible schemata $\phi_1, \ldots, \phi_k$ is finite, and thus the set of possible schemata ϕ is finite up to contraction. □

We write $\psi \sqsubset \phi$ if ϕ is of the form $\psi \wedge \psi'$ (modulo the AC property of $\wedge$). Intuitively, a conjunction $\psi_1 \wedge \cdots \wedge \psi_k$ is seen as a set – as it is the case in the tableau calculus considered here – and $\sqsubset$ denotes the membership of this set.

Proposition 8. *Let ψ be a schema in $\mathcal{E}_{\rho^* \circ unfold}(\phi)$. Let $\Pi^{\alpha}_{\mathtt{i}=0}\zeta$ be an iteration in ψ (where $\Pi \in \{\bigwedge, \bigvee\}$). Then the following conditions hold:*

1. *If $\psi \neq \phi$ then $\alpha = \mathtt{n}$.*
2. *ζ occurs in ϕ.*
3. *If $\phi \neq \psi$ and ζ contains $\mathtt{i}+1$ then $\Pi^{\mathtt{n}}_{\mathtt{i}=0}\zeta \sqsubset \psi$.*
4. *If $\phi \neq \psi$ and ζ does not contain $\mathtt{i}+1$ then $\Pi^{\mathtt{n}}_{\mathtt{i}=0}\zeta$ occurs in a schema of the form $\Pi^{\mathtt{n}}_{\mathtt{i}=0}\zeta\, \pi\, \zeta\{\mathtt{i} \leftarrow \mathtt{n}+1\}$ in ψ, where $\pi \in \{\vee, \wedge\}$.*
5. *If $\phi \neq \psi$, $\psi \notin \rho^*(unfold(\phi\{\mathtt{n} \leftarrow \mathtt{n}+1\}))$ and ζ does not contain $\mathtt{i}+1$ then $\Pi^{\mathtt{n}}_{\mathtt{i}=0}\zeta\, \pi\, \zeta\{\mathtt{i} \leftarrow \mathtt{n}+1\} \sqsubset \psi$.*
6. *If $\lambda \sqsubset \psi$ and λ contains no iteration then* $\mathrm{depth}(\lambda) \leq \mathrm{depth}(\phi)$.

Proof. The proof is by induction on the construction of the set $\mathcal{E}_{\rho^* \circ unfold}(\phi)$. It is straightforward if $\phi = \psi$. Otherwise, by definition of $\mathcal{E}_{\rho^* \circ unfold}(\phi)$, ψ occurs in a set $\rho^*(unfold(\psi'\{\mathtt{n} \leftarrow \mathtt{n}+1\}))$, where ψ' is some schema in $\mathcal{E}_{\rho^* \circ unfold}(\phi)$ (with possibly $\psi' = \phi$).

Since the rules in ρ^* do not introduce new iterations, $\Pi^{\alpha}_{\mathtt{i}=0}\zeta$ must occur in $unfold(\psi'\{\mathtt{n} \leftarrow \mathtt{n}+1\})$. This implies that the iteration is irreducible by unfolding hence $\alpha = \mathtt{n}$ (1). Furthermore, $\Pi^{\alpha}_{\mathtt{i}=0}\zeta$ cannot occur in a schema of the form $\psi'\{\mathtt{n} \leftarrow \mathtt{n}+1\}$ (precisely because $\alpha = \mathtt{n}$) hence it must have been generated by unfolding. Thus it occurs in a schema of the form $(\Pi^{\mathtt{n}}_{\mathtt{i}=0}\zeta)\pi(\zeta\{\mathtt{i} \leftarrow \mathtt{n}+1\})$ in $unfold(\psi'\{\mathtt{n} \leftarrow \mathtt{n}+1\})$ (where $\pi = \vee$ if $\Pi = \bigvee$ and $\pi = \wedge$ if $\Pi = \bigwedge$). Then necessarily an iteration of the form $\Pi^{\mathtt{n}+k}_{\mathtt{i}=0}\zeta$ must occur in ψ'. By the induction hypothesis, ζ occur in ϕ (2).

Assume that ζ contains $\mathtt{i}+1$. Then $(\Pi^{\mathtt{n}}_{\mathtt{i}=0}\zeta)\pi(\zeta\{\mathtt{i} \leftarrow \mathtt{n}+1\})$ is necessarily non-$\mathtt{n}$-elementary. Hence it must be decomposed by the rules in ρ and we must have $\Pi^{\mathtt{n}}_{\mathtt{i}=0}\zeta \sqsubset \psi$ (3). Now, assume that ζ does not contain $\mathtt{i}+1$. Then $(\Pi^{\mathtt{n}}_{\mathtt{i}=0}\zeta)\pi(\zeta\{\mathtt{i} \leftarrow \mathtt{n}+1\})$ in $unfold(\psi'\{\mathtt{n} \leftarrow \mathtt{n}+1\})$ is $\mathtt{n}$-elementary, hence it cannot be decomposed by the rules in ρ. Therefore, it must occur in ψ (4).

Assume that $\psi \notin \rho^*(unfold(\phi\{\mathtt{n} \leftarrow \mathtt{n}+1\}))$ and that ζ does not contain $\mathtt{i}+1$. By the induction hypothesis, since $\psi' \neq \phi$, $\Pi^{\mathtt{n}}_{\mathtt{i}=0}\zeta$ occurs in a schema $\Pi^{\mathtt{n}}_{\mathtt{i}=0}\zeta\pi(\zeta\{\mathtt{i} \leftarrow \mathtt{n}+1\})$ in ψ'. Then $\psi'\{\mathtt{n} \leftarrow \mathtt{n}+1\}$ contains $\Pi^{\mathtt{n}+1}_{\mathtt{i}=0}\zeta\pi(\zeta\{\mathtt{i} \leftarrow \mathtt{n}+2\})$ which is reduced by unfolding to $\Pi^{\mathtt{n}}_{\mathtt{i}=0}\zeta\pi(\zeta\{\mathtt{i} \leftarrow \mathtt{n}+1\})\pi(\zeta\{\mathtt{i} \leftarrow \mathtt{n}+2\})$. $\zeta\{\mathtt{i} \leftarrow \mathtt{n}+2\}$ is non-$\mathtt{n}$-elementary (since ζ contains $\mathtt{i}$) hence the decomposition rules in ρ apply on the previous schema and we must have $\Pi^{\mathtt{n}}_{\mathtt{i}=0}\zeta\pi(\zeta\{\mathtt{i} \leftarrow \mathtt{n}+1\}) \sqsubset \psi$ (5).

Finally, let $\lambda \sqsubset \psi$ be an iteration-free schema. Since the rules in ρ cannot increase the depth of the schema, there exists an iteration-free schema $\lambda' \sqsubset unfold(\psi'\{\mathtt{n} \leftarrow \mathtt{n}+1\})$ such that $\mathrm{depth}(\lambda') \geq \mathrm{depth}(\lambda)$. Then, since λ' contains no iteration it either occurs in $\psi'\{\mathtt{n} \leftarrow \mathtt{n}+1\}$ or in a schema $\zeta\{\mathtt{i} \leftarrow \mathtt{n}+1\}$, where ζ is an iteration body in ψ'. In both cases we have obviously $\mathrm{depth}(\lambda') \leq \mathrm{depth}(\phi)$ (by induction hypothesis) thus $\mathrm{depth}(\lambda) \leq \mathrm{depth}(\phi)$ (6). □

Let ψ be a schema in $\mathcal{E}_{\rho^* \circ unfold}(\phi)$ distinct from the ones in $\{\phi\} \cup \rho^*(unfold(\phi\{\mathtt{n} \leftarrow \mathtt{n}+1\}))$. ψ can be written in the form $\psi_1 \wedge \cdots \wedge \psi_k$ where the $\psi_1, \ldots, \psi_k$ are not conjunctions. If some ψ_l (for $l \in [1,k]$) contains an iteration then by the previous proposition (5) it is either an iteration in ϕ or of the form $\Pi^{\mathtt{n}}_{\mathtt{i}=0}\zeta\pi(\zeta\{\mathtt{i} \leftarrow \mathtt{n}+1\})$ where $\Pi^{\mathtt{n}}_{\mathtt{i}=0}\zeta$ is an iteration in ϕ. This implies that the depth of ψ_l is bounded by $\mathrm{depth}(\phi)+1$ (since the depth of $\Pi^{\mathtt{n}}_{\mathtt{i}=0}\zeta$ is lower or equal to the one of ϕ). If ψ_l is not an iteration, then by the previous proposition (6) we know that $\mathrm{depth}(\psi_l) \leq \mathrm{depth}(\phi)$. Thus by Proposition 7 we conclude that the set $\mathcal{E}_{\rho^* \circ unfold}(\phi)$ is finite (up to contraction).

G Proof of Theorem 4

Lemma 5. *Let $\lambda_1, \ldots, \lambda_k$ be a set of ground literals such that, for every $l \in [1,k]$, $Ind(\lambda_l)$ is of the form $\{\mathtt{n}+m+1\}$, for some $m \in \mathbb{N}$ (possibly depending on l). If $\lambda_1 \wedge \cdots \wedge \lambda_k$ is $\mathcal{T}$-satisfiable then $(\lambda_1 \wedge \cdots \wedge \lambda_k)\{\mathtt{n} \leftarrow l\}$ is $\mathcal{T}$-satisfiable, for any $l \in \mathbb{N}$.*

Proof. Let $I \in \mathcal{T}$ be a model of $\lambda_1 \wedge \cdots \wedge \lambda_k$. We obtain a model of $(\lambda_1 \wedge \cdots \wedge \lambda_k)\{\mathtt{n} \leftarrow l+1\}$ by interpreting every indexed symbol f_m such that $m \geq l$ as $\langle f_{m-l+\langle \mathtt{n} \rangle^I} \rangle^I$ (since the indexed symbols are non-built-in, the obtained interpretation is still in $\mathcal{T}$). Notice that the interpretation of f_0 is not affected. It is easy to prove that for every expression ψ occurring in $\lambda_1 \wedge \cdots \wedge \lambda_k$, we have $\langle \psi \rangle^I = \langle \psi\{\mathtt{n} \leftarrow l+1\} \rangle^J$ (by a straightforward induction on ψ). □

Let I be an interpretation satisfying $\Delta(\lambda_1 \wedge \cdots \wedge \lambda_k)$. Then we must have $\Delta(\lambda_1 \wedge \cdots \wedge \lambda_k) = \mathtt{true}$ hence $\lambda_1 \wedge \cdots \wedge \lambda_k$ is $\mathcal{T}$-satisfiable. Let $J \in \mathcal{T}$ such that $J \models \lambda_1 \wedge \cdots \wedge \lambda_k$. By Lemma 5, we can assume that $\langle \mathtt{n} \rangle^I = \langle \mathtt{n} \rangle^J$. Since $\lambda_1, \ldots, \lambda_k$ are non-$\mathtt{n}$-elementary, for every $i \in [1,k]$, $Ind(\lambda_i)$ contains an expression of the form $\mathtt{n}+l$ where $l > 1$. But then, since $\mathfrak{L}$ is homogeneous, every index in $Ind(\lambda_i)$ must have this property. Thus the truth value of λ_i in J only depends on the interpretation of the indexed symbols f_α where $\alpha > \langle \mathtt{n} \rangle^J + 1$. Thus we can assume that J coincides with I on every indexed symbol f_α where $\alpha \leq \langle \mathtt{n} \rangle^J + 1$. Then we have $J \sim^{\mathfrak{L}_\diamond} I$.

Now, assume that there exists an interpretation $J \sim^{\mathfrak{L}_\diamond} I$ such that $J \models \lambda_1 \wedge \cdots \wedge \lambda_k$. By Lemma 5, $(\lambda_1 \wedge \cdots \wedge \lambda_k)\{\mathtt{n} \leftarrow 0\}$ is $\mathcal{T}$-satisfiable. Then by definition $\Delta(\lambda_1 \wedge \cdots \wedge \lambda_k) = \mathtt{true}$ and obviously $I \models \Delta(\lambda_1 \wedge \cdots \wedge \lambda_k)$.

H Proof of Theorem 5

Lemma 6. *Let I, J be two interpretations such that $I \sim^{\mathfrak{L}_\preceq} J$. If λ is a ground $\mathtt{n}$-elementary literal in $\mathfrak{L}_\preceq$ then $\langle \lambda \rangle^I = \langle \lambda \rangle^J$.*

Proof. By inspection of the different cases in the definition of $\mathfrak{L}_\preceq$, it is easy to see that λ must be of the form $\lambda'\{\mathtt{i} \leftarrow \mathtt{n}\}$, for some $\lambda' \in \mathfrak{L}_\preceq$ (since λ contains no occurrence of $\mathtt{n}+2$). Then by definition of $\sim^{\mathfrak{L}_\preceq}$ we deduce that $\forall k \in [0, \langle \mathtt{n} \rangle^I], \langle \lambda'\{\mathtt{i} \leftarrow k\} \rangle^I = \langle \lambda'\{\mathtt{i} \leftarrow k\} \rangle^J$ hence in particular (for $k = \langle \mathtt{n} \rangle^I$) that $\langle \lambda \rangle^I = \langle \lambda \rangle^J$. □

Let ϕ be a conjunction of ground non-$\mathtt{n}$-elementary literals in $\mathfrak{L}_{\preceq}$. Let I be an interpretation. Assume that there exists an interpretation $J \sim^{\mathfrak{L}_{\preceq}} I$ satisfying ϕ. Then for every pair of literals $a_{\mathtt{n}+2} \approx u$ and $a_{\mathtt{n}+2} \approx v$ occurring in ϕ we must have $J \models u \approx v$, thus $J \models \tau(u \approx v)$. Since by definition every literal in $\tau(u \approx v)$ is $\mathtt{n}$-elementary and occurs in $\mathfrak{L}_{\preceq}$, by Lemma 6, I and J coincide on $\tau(u \approx v)$ hence $I \models \tau(u \approx v)$. Finally, if $\phi \models u \preceq v$ and if $u \preceq v$ is $\mathtt{n}$-elementary and is in $\mathfrak{L}_{\preceq}$, then we must have $J \models u \preceq v$ (since $J \models \phi$) and by Lemma 6: $I \models \tau(u \preceq v)$. Thus $I \models \Delta_{\preceq}(\phi)$.

Now, assume that $I \models \Delta_{\preceq}(\phi)$. Let $\geq$ be the interpretation of $\succeq$ in I. W.l.o.g. we assume that for every constant symbol $a \in \mathcal{C}_{\preceq}$ there exists at least one term m^a without index such that $a_{\mathtt{n}+2} \succeq m^a$ occurs in ϕ. If it is not the case, it suffices to add to ϕ, for every $a \in \mathcal{C}_{\preceq}$, the literal $a_{\mathtt{n}+2} \succeq m^a$, where m^a is a fresh non-indexed constant symbol of the same sort as a. Then the interpretation I is extended by interpreting m^a as an arbitrary value lower or equal to all other ground terms of the same sort in I.

Let J be the interpretation coinciding with I except for the interpretation of the indexed constants of the form $a_{\langle \mathtt{n} \rangle^I + 2}$ where $a \in \mathcal{C}_{\approx} \cup \mathcal{C}_{\preceq}$, that are defined as follows:

- If $a \in \mathcal{C}_{\preceq}$ then $\langle a_{\mathtt{n}+2} \rangle^J$ is the maximal (according to the ordering $\leq$) value $\langle u \rangle^I$ such that u is an $\mathtt{n}$-elementary term with $\phi \models a_{\mathtt{n}+2} \succeq u$.
- If $a \in \mathcal{C}_{\approx}$ and ϕ contains a literal $a_{\mathtt{n}+2} \approx u$ then $\langle a_{\mathtt{n}+2} \rangle^J \stackrel{\text{def}}{=} \langle u \rangle^I$. Since $I \models \Delta_{\preceq}(\phi)$, we have $I \models E(\phi)$ (by definition of $\Delta_{\preceq}$) hence for every pair of literals $a_{\mathtt{n}+2} \approx u$ and $a_{\mathtt{n}+2} \approx v$ occurring in ϕ we must have $I \models \tau(u \approx v) \equiv_{\mathcal{T}} u \approx v$. Therefore the interpretation of $a_{\mathtt{n}+2}$ does not depend on the choice of u. If no such term u exists the interpretation of $a_{\mathtt{n}+2}$ is arbitrary.

By construction, it is clear that $I \sim^{\mathfrak{L}_{\preceq}} J$ (since I and J coincide on every symbol whose index is in $[0, \langle \mathtt{n} \rangle^I + 1]$). Now we prove that $J \models \phi$. Let λ be a literal in ϕ. We distinguish several cases, according to the form of λ (see the definition of $\mathfrak{L}_{\preceq}$):

- λ is of the form $a_{\mathtt{n}+2} \succeq u$, where u is an $\mathtt{n}$-elementary term. We have $\phi \models a_{\mathtt{n}+2} \succeq u$, hence $\langle a_{\mathtt{n}+2} \rangle^J \succeq \langle u \rangle^I = \langle u \rangle^J$. Thus $J \models \lambda$.
- λ is of the form $a_{\mathtt{n}+2} \succeq b_{\mathtt{n}+2}$, where $a, b \in \mathcal{C}_{\preceq}$. We have $\phi \models b_{\mathtt{n}+2} \succeq u \Rightarrow \phi \models a_{\mathtt{n}+2} \succeq u$, hence by definition $\langle a_{\mathtt{n}+2} \rangle^J \geq \langle b_{\mathtt{n}+2} \rangle^J$. Thus $J \models \lambda$.
- λ is of the form $a_{\mathtt{n}+2} \preceq u$, where u is an $\mathtt{n}$-elementary term. By definition, there exists an $\mathtt{n}$-elementary term v such that $\langle a_{\mathtt{n}+2} \rangle^J = \langle v \rangle^I = \langle v \rangle^J$ and $\phi \models a_{\mathtt{n}+2} \succeq v$. Then we must have by transitivity $\phi \models v \preceq u$. u and v are both $\mathtt{n}$-elementary. Moreover, by definition of $\mathfrak{L}_{\preceq}$ the inequation $v \preceq u$ must occur in $\mathfrak{L}_{\preceq}$. Thus $E(\phi)$ contains the literal $v \preceq u$, by definition of $\Delta_{\preceq}$. Consequently we have $I \models v \preceq u$, hence $\langle v \rangle^I \leq \langle u \rangle^I$. Since I and J coincide on $\mathtt{n}$-elementary terms, this implies that $J \models \lambda$.
- λ is of the form $a_{\mathtt{n}+2} \approx u$. The proof is immediate by construction of J.